IMPROVED SAMPLING BASED MOTION PLANNING THROUGH LOCAL LEARNING

A Dissertation

by

CHINWE PAMELA EKENNA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Nancy M. Amato
Committee Members,	J. Martin Scholtz
	Dezhen Song
	Tiffani L. Williams
Head of Department,	Dilma Da Silva

August 2016

Major Subject: Computer Science

ABSTRACT

Every motion made by a moving object is either planned implicitly, e.g., human natural movement from one point to another, or explicitly, e.g., pre-planned information about where a robot should move in a room to effectively avoid colliding with obstacles. Motion planning is a well studied concept in robotics and it involves moving an object from a start to goal configuration. Motion planning arises in many application domains such as robotics, computer animation (digital actors), intelligent CAD (virtual prototyping and training) and even computational biology (protein folding and drug design). Interestingly, a single class of planners, sampling-based planners have proven effective in all these domains.

Probabilistic Roadmap Methods (PRMs) are one type of sampling-based planners that sample robot configurations (nodes) and connect them via viable local paths (edges) to form a roadmap containing representative feasible trajectories. The roadmap is then queried to find solution paths between start and goal configurations. Different PRM strategies perform differently given different input parameters, e.g., workspace environments and robot definitions.

Motion planing, however, is computationally hard – it requires geometric path planning which has been shown to be PSPACE hard, complex representational issues for robots with known physical, geometric and temporal constraints, and challenging mapping/representing requirements for the workspace environment. Many important environments, e.g., houses, factories and airports, are heterogeneous, i.e., contain free, cluttered and narrow spaces. Heterogeneous environments, however, introduce a new set of problems for motion planing and PRM strategies because

there is no ideal method suitable for all regions in the environment.

In this work we introduce a technique that can adapt and apply PRM methods suitable for local regions in an environment. The basic strategy is to first identify a local region of the environment suitable for the current action based on identified neighbors. Next, based on past performance of methods in this region, adapt and pick a method to use at this time. This selection and adaptation is done by applying machine learning.

By performing the local region creation in this dynamic fashion, we remove the need to explicitly partition the environment as was done in previous methods and which is difficult to do, slows down performance and includes the difficult process of determining what strategy to use even after making an explicit partitioning. Our method handles and removes these overheads.

We show benefits of this approach in both planning robot motions and in protein folding simulations. We perform experiments on robots in simulation with different degrees of freedom and varying levels of heterogeneity in the environment and show an improvement in performance when our local learning method is applied. Protein folding simulations were performed on 23 proteins and we note an improvement in the quality of pathways produced with comparable performance in terms of time needed to build the roadmap.

DEDICATION

God, giving me the strength to finish this journey successfully.

My Dad (In Memory) "Pay attention to details" really paid off. Thank you Sir.

My Mum "When the time of the fruit comes, it will surely fall".

My Siblings, giving me something to laugh about irrespective of situations.

My Husband Obioma Obim, you make the journey worth it. I love you.

ACKNOWLEDGEMENT

I would first like to thank my advisor, Dr. Nancy Amato, for mentoring me through the years and teaching me how to get out of my comfort zone. I truly appreciate learning under you.

I would also like to thank my committee members, Dr. J. Martin Scholtz, Dr. Dezhen Song, and Dr. Tiffani L. Williams, for their support and feedback. I appreciate the time taken to help me scale through and streamline my research productively.

I would like to thank Dr. Shawna Thomas for taking out time to teach and explain things to me patiently and her interest in my success both personally and academically.

I also thank all the collaborators that I have worked with over the years, both on this work and on other research projects: Diane Uwacu, Hsin(Cindy) Yeh, Mukulika Ghosh, and Jory Denny. I am grateful to have interacted with Cindy and Mukulika through out my graduate studies, striving, encouraging each other and attending conferences together, we had a lot of laughs together. Thank you to all the Parasol members, both former and current.

Thanks to the Schlumberger Faculty for the Future Fellowship for their support during my studies. I would also like to thank the conferences and workshops that provides travel grants for me to participate and present my research findings. These include funding to attend the Grace Hopper Celebration of Women in Computing Conference, the Richard Tapia Celebration of Diversity in Computing, the IEEE/RSJ International Conference on Intelligent Robots and Systems, the IEEE International Conference on Robotics and Automation, and the IEEE International Conference on

Bioinformatics and Biomedicine.

I would like to thank my family for their steady support. My dad (in memory) who was steadfast in seeing me as worthy of the best and sacrificing to make that happen. My mum, who has sacrificed right from the beginning, time, money, prayers, love, thank you mummy for being your humble self "The time of nnamma has passed indeed". My brother Uchenna (in memory), you would have been proud of your little sister. My siblings Nnenna, Ike, Ijeoma, Oke, thank you for making the time fly by with funny anecdotes, words of encouragement and white noise.

Finally, I would like to specially thank my husband Obioma Nwachukwu, your faith in my abilities is astounding, thank you for being patient with me and loving me through it all, I love you dear.

TABLE OF CONTENTS

Page

LIST OF FIGURES

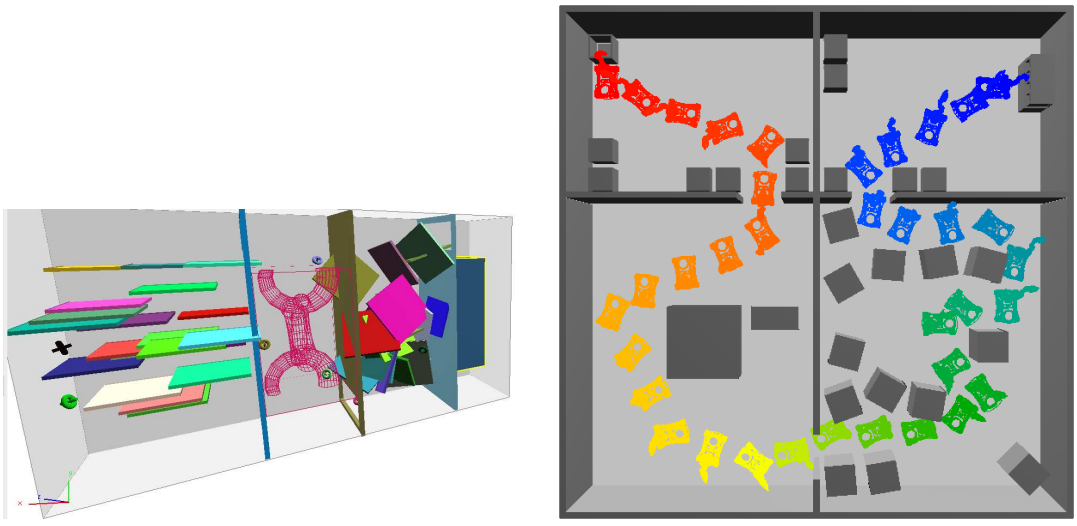
FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Planning motions is needed in many disciplines such as planning for deformable robots [42,86,94], manipulation planning [55], computational biology search problems [81, 102], character animation for games and movies, and virtual prototyping. A motion planner finds a valid sequence of motions, or path, for a robot to transit from an initial state to a goal state or reports that no such path exists. The robot can be any movable system: an articulated arm in a factory, a car, or a protein. Robots or objects most often have to plan and navigate in heterogeneous environments, i.e., containing a combination of free spaces, narrow tunnels and obstacles. Environments as shown in Figure 1.1 are heterogeneous because robots at different positions would have varying visibility of the entire space. These robots also have different complexity ranging from rigid bodies to highly articulated linkages with many degrees of freedom.



(a) 3D Maze Heterogeneous (b) Heterogeneous Room Setting

Figure 1.1: Heterogeneous environments.

Various motion planning methods have been developed to solve this problem but no single planner is scalable to all environments and this could be due to a number of reasons. One reason is that the heterogeneity of the environment is not investigated and so the environment is seen as homogeneous [16, 29] which our research shows is not always the case and affects performance. Another reason is the concentration of research to a specific robot type and type of environment which limits the use of these methods across other environments [2, 8, 52].

Previous work looked into explicitly partitioning the environment and then applying different methods in the different partitions. However, this is difficult to do due to the challenge of identifying when the environment has been broken into homogeneous pieces or determining if the right method has been applied [83, 100]. Our method addresses this issue by dynamically creating regions and uses machine learning techniques to select the appropriate methods to apply based on information regarding their past performance in that local region. Thus we remove the need to explicitly partition or know beforehand what method is suitable in different parts of the environment.

1.1 Research Contributions

The goal of this research is to develop a means to dynamically create regions in heterogeneous environments, access stored information about the performance of methods in these identified regions, and then intelligently decide what method is most suitable during the current iteration. This research focuses on Probabilistic Roadmap Methods (PRMs) [58]. PRMs are state of the art motion planning algorithms that solve motion planning problems in two phases. During the sampling stage, valid configurations of the robot in the environment are generated, and dur-

ing the connection stage those sampled nodes are connected together with edges to construct a roadmap that is used to find the valid path. We study characteristics peculiar to different probabilistic planners in a bid to utilize their usefulness when needed in these different motion planning scenarios.

We utilize reinforcement learning approaches to intelligently decide which PRM method to apply in the local region and the current iteration during roadmap construction. Our technique extends different algorithms such as the multi-arm bandit problem [11, 20, 99]. We include a localizing feature that keeps the learning sensitive to regions in the environment with similar characteristics, and which enables learning from past experience of the methods in these regions so that methods most suitable for the current iteration can be selected. By applying learning in these dynamically determined regions, we remove the need to explicitly partition environments and the overhead of deciding which method to use for a given input problem.

Our results show that we either achieve improved performance or at the least comparable performance with the best single planning method. We test on a variety of heterogeneous environments and study the performance of our learning based approach. We compare the performance of local learning to global learning (no local region identification).

Our results show our framework is able to make improvements on the roadmap quality and to solve the query problems in less time than other non-learned scenarios in most cases.

The main contributions of this research include the following:

- A local region feature created on the fly that localizes learning to areas of interest and based on past performance applies suitable methods in the current

locale and iteration.

- A machine learning reward based technique that locally rewards the performance of these methods.
- A technique that utilizes the strengths of these PRM methods while minimizing their weaknesses (use when and where needed).

Portions of this research have been previously published and presented. The basic application of learning to PRM was presented at *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* in 2013 [35]. Improvements on the learning strategy and introduction of the local region feature was presented at *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* in 2015 [37]. A discussion and investigation on the application of local learning to the different phases of PRM roadmap construction was presented at *The Machine Learning in Planning and Control of Robot Motion Workshop (IROS-MLPC)* in 2015 [39]. Application of local learning to protein folding was presented at *The IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* in 2015 [36].

1.2 Outline

Chapter 2 discusses some important primitives and background on motion planning methods and the multi-arm bandit problem algorithm, and then discusses some related work on sampling methods, connection methods, and adaptive learning strategies employed to solve the motion planning problem. Chapter 3 describes our learning framework and the algorithms developed. Chapter 4 discusses our local learning approach showing experiments we performed when applied to sampling and connection separately and then investigates what happens when applied to both sampling and connection. Chapter 5 shows an application of local learning to proteins, studying

different protein folding simulations and the application of our method to the sampling and connection stages. We finally conclude and discuss future work in Chapter 6.

2. PRELIMINARIES AND RELATED WORK

In this section we discuss some motion planning primitives, and sampling based motion planning algorithms including graph based and tree based methods. We further describe reinforcement learning techniques applicable to our framework, PRMs, existing sampling methods, existing connection methods and machine learning strategies applied to motion planning.

2.1 Motion Planning

The motion planning problem involves finding a valid path (e.g., collision-free and satisfying all joint limit and/or loop closure constraints) for a movable object starting from its start configuration to a goal configuration in an environment [24]. A single configuration is defined based on the movable object's d independent parameters or degrees of freedom (DOF). The set of all possible configurations (both feasible and infeasible) defines a configuration space (C-Space) [24, 91]. C-Space is partitioned into two components: C-free (the set of all feasible configurations) and C-obst (the set of all infeasible configurations). Motion planning then becomes the problem of finding a continuous sequence of points in C-free that connects the start and the goal configuration.

A complete solution to the motion planning problem is known to be computationally expensive and it has been shown that this problem is PSPACE-hard with an upper bound that is exponential in the number of the movable object's DOFs [24, 91]. Basically, any planner that is guaranteed to find a solution or determine that none exists requires exponential space that is in the the number of DOFs. Heuristic and approximate algorithms were therefore implemented that trade completeness for efficiency

and sampling-based motion planning is one such approach.

2.1.1 Sampling Based Motion Planning

Sampling-based methods [24] are a state-of-the-art approach to solving motion planning problems. These methods are known to be probabilistically complete because even though there is no guarantee to find a solution if one exists, the probability of finding a solution if it exists increases as the number of samples generated also increases. Sampling-based methods are broadly classified into two main classes: graph-based methods such as the Probabilistic Roadmap Method (PRM) [58] and tree-based methods such as Expansive-Space tree planner (ESTs) [47] and Rapidly-exploring Random Tree (RRT) [65].

2.1.1.1 Graph-Based Methods

Probabilistic RoadMaps (PRMs) [58] are sampling-based motion planning approaches that use a two stage process to solve planning problems: roadmap construction and query processing. During roadmap construction, PRMs sample the configuration space (C-Space), i.e., the set of all possible robot placements, retaining valid ones as roadmap nodes, and attempting to connect them using some local planner (e.g., straightline). PRM solves motion planning problems by constructing a graph $G = (V, E)$, called a roadmap, to show how connected C-free is (Figure 2.1 [3]). PRMs have been shown to be probabilistically complete [57].

The basic PRM [58] (shown in Algorithm 1), begins with generating nodes using uniform random sampling and then attempts connections between a node and its k-nearest neighbors as computed using some distance metric (e.g., Grid based, Root-Mean-Square or Euclidean distance [5]). Once the roadmap is constructed, query processing is done by connecting the start and goal configurations to the roadmap

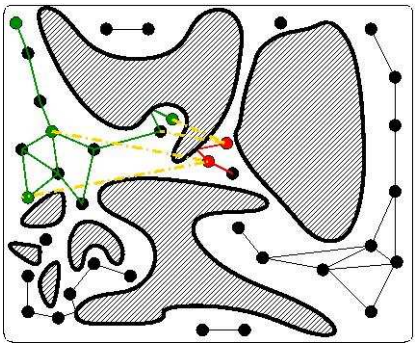


Figure 2.1: An illustration of PRM*

and extracting a path from the roadmap that connects them. Many variants of PRMs have been proposed that bias node generation or connection or query processing in various ways to handle instances such as narrow corridors or obstacles too close to the boundary; we discuss some of these variants in Section 2.3.

2.1.1.2 Tree-Based Methods

The Expansive Space Tree Planner (ESTs) and Rapidly-exploring Random Tree (RRT) are two common tree based methods in sampling based motion planning. These methods are most commonly used in solving single query problems. Specifically, RRT is particularly well suited for non-holonomic and kinodynamic motion planning problems [66, 67]. RRT (shown in Algorithm 2 and illustrated in Figure 2.2 [3]) grows a tree rooted at the start configuration and expands towards unexplored areas of the C-Space. RRT begins by generating a uniform random sample q_{rand}, and identifies the closest node q_{near} in the tree to q_{rand}, and then q_{near} is "extended" toward q_{rand} for a stepsize of at most Δq. If the extension is successful, q_{new} is added to the tree as a node and the pair q_{near} and q_{new} is added as an edge.

*Reprinted with permission from "Improving the Connectivitiy of PRM Roadmaps" by Marco Morales, Samuel Rodriguez, Nancy M. Amato, 2003 IEEE International Conference on Robotics and Automation (ICRA), pp. 4427-4432 [82] ©2003 IEEE.

Algorithm 1 Basic PRM

Input. An environment env, number of nodes N
Output. A roadmap graph G containing N valid nodes connected

```
1: i ← 0
2: while i < N do
3:    q ← GetRandomValidNode(env)
4:    G.AddNode(q)
5:    i ← i + 1
6: end while
7: for all q ∈ G do
8:    Q ← FindNeighbor(G, q, k)
9:    for all q_near ∈ Q do
10:      if local planner can connect q and q_near then
11:         G.AddEdge(q, q_near)
12:      end if
13:   end for
14: end for
15: return G
```

To solve a particular query, RRT repeats this process until the goal configuration is connected to the tree. RRT-connect is a variant that grows two trees towards each other; one rooted at the start configuration and the other at the goal configuration [59]. These two trees explore C-Space until they are connected. Many variants of RRT have been proposed and discussed [15, 24, 32, 51, 56, 80, 93].

2.1.2 Heterogeneity of the C-Space

Dale and Amato in [28] made some interesting analysis in defining the heterogeneity of a space. They defined four characterizing measures to help determine when a region of the C-Space is heterogeneous. They classify a region of the C-Space based on the ratio of non-colliding nodes to all nodes sampled. If this ratio is one, then the

[†]Reprinted with permission from "Adapting RRT Growth for Heterogeneous Environments" by Jory Denny, Marco A. Morales A., Samuel Rodriguez, Nancy M. Amato, 2013 IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 1772 - 1778 [33] ©2013 IEEE.

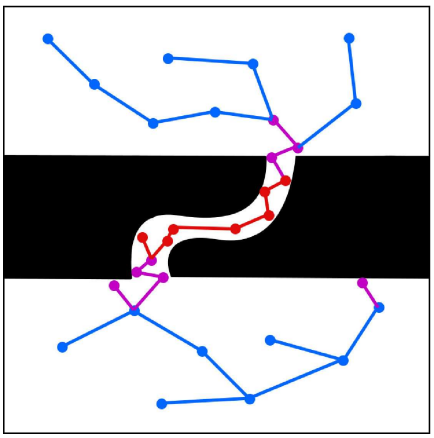


Figure 2.2: An illustration of RRT†

Algorithm 2 Basic RRT

Input. An environment env, a root q_{root}, the number of nodes N
Output. A tree T containing N nodes rooted at q_{root}

```
1: T.AddNode(q_root)
2: i ← 0
3: while i < N do
4:    q_rand ← GetRandomNode(env)
5:    q_near ← FindNeighbor(T, q_rand, 1)
6:    q_new ← Extend(q_near, q_rand)
7:    if !TooSimilar(q_near, q_new) ∧ IsValid(q_new) then
8:       T.AddNode(q_new)
9:       T.AddEdge(q_near, q_new)
10:      i ← i + 1
11:   end if
12: end while
13: return T
```

region is **free**, C-free . If the ratio equals zero, then the region is **blocked**, C-obst.

Another metric they used to define the workspace is based on the number of connected components and the number of nodes present (size) in a connected component (CC). Figure 2.3 shows an example environment with a mixture of free and cluttered

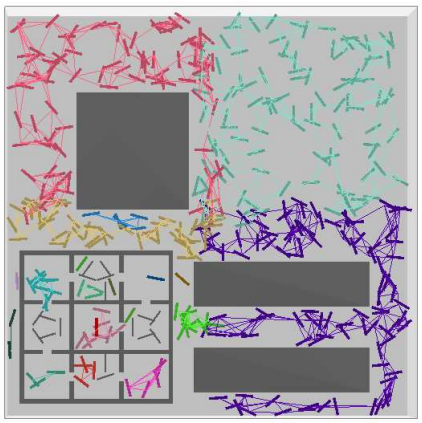


Figure 2.3: Heterogeneity

regions. Using the ratio method it becomes hard to make an informed decision about the C-Space and how heterogeneous it is. In a free space, the CC will be 1 and the size of the CC will include all valid nodes. As the number of CC increases and the size of the CC reduce then we identify more cluttered regions. This difference helps characterize the right top half of the figure as free and the remaining as **cluttered**.

The final metric they discuss relates to defining obstacles in the workspace. They determine an **obstacle** based on the ratio of surface connection and surface connections attempted as seen in Figure 2.3. This inclusion of the surface was based on research in [8, 17] that observed that given that nodes can be obtained arbitrarily close to configuration obstacle surfaces, the results of connection attempts between nodes near the surface of an obstacle can be used to provide a surface characterization for the obstacle.

Given this characterization, we define a heterogeneous C-Space as containing more than one type of region – free space, cluttered or obstacle regions.

2.2 Reinforcement Learning

Reinforcement learning is a machine learning concept that involves learning what actions to take in a given scenario to maximize a cumulative reward. This learning concept focuses on utilizing exploitation and exploration techniques.

Exploitation involves continually using the best available methods while exploration continually looks for better methods from the list that could potentially be exploited. Balancing when to explore or exploit is an open research question that has been studied for decades. In our work we utilize one of the standard approaches called the multi-arm bandit problem.

2.2.1 Multi-arm Bandit Problem

In this section we describe the reinforcement learning approach called the multi-arm bandit problem which is an important primitive we employ in our work.

The Multi-armed bandit problem (MAB) was first presented by Robbins [63] and provided a platform for modeling how automated agents gain new knowledge when exploring their environment. This is done by exploiting currently reliable knowledge about the environment at a particular point in time. MAB is a reinforcement learning strategy that investigates the trade-off between exploration and exploitation.

A common analogy of the MAB problem is that of a gambler with multiple slot machines, the gambler keeps playing the hand that is winning (exploitation) while looking out for other slots when they become a winning hand.

As a stochastic approach, the bandit problem consists of a set of K probability distributions $(P_1, ..., P_K)$ with associated expected values $(\varphi_1, ..., \varphi_K)$ and variances $(\sigma_1^2, ..., \sigma_k^2)$. Initially, the P_i are unknown to the player.

These probability distributions generally correspond to the arms for a slot machine and the player is viewed as a gambler with a goal of winning as much money as possible by pulling all these arms as much as possible.

At each iteration, $t = 1, 2, ...$, the player selects an arm, with index j(t), and receives a reward $r(t) \backsim P_j(t)$. The player has a two-fold goal: on one hand, identify quickly the winning hand; on the other hand, improve on the rewards collected as much as possible while playing. Bandit algorithms specify a strategy by which the player should choose an arm j(t) at each turn [61].

A popular performance measure for bandit algorithms is the total expected regret, the regret is how much worse the algorithm performs as opposed to the best experts decisions.

The regret T is calculated as: $R_T = T\varphi^* - \sum_{t=1}^{T} \varphi_j(t)$ where $\varphi^* = max_{i=1,...k}\varphi^1$ is the expected reward from the best arm [20].

The payoff of the algorithm at step t is defined as the number of correct guesses minus the number of wrong guesses.

2.3 Related Work

In this section we discuss relevant work that has been done in the field of sampling based planning both during sampling and connection. We include discussions about adaptive methods that have been employed in all these cases as well.

2.3.1 Existing Sampling Methods

Considerable effort has been dedicated to finding ways of increasing sampling in narrow and difficult regions of environments. In this section we look at some of the most successful strategies and their individual strengths and limitations.

- Uniform

    1. Uniform Sampling [58]: This method generates nodes uniformly at random in C-Space retaining valid ones. A drawback with this approach is its inability to sample narrow passages efficiently thus increasing the chances of oversampling in open areas.

- Near Obstacles

    1. Obstacle-Based PRM (OBPRM) [7, 8]: samples configurations near C-obst surfaces by pushing configurations to the C-obst boundary. Even if OBPRM excels in narrow passages, it can be expensive because it requires many validity tests. In addition, the nature of its sampling tends to produce paths with low clearances.
    2. Gaussian PRM [17]: This technique attempts to generate configurations that are a Gaussian distance d away from the obstacle surfaces. A first configuration is randomly generated and the second one is generated a Gaussian distance d away from the first configuration, where d is a user-specified parameter. If the validity of the two configurations differ, the valid one will be retained as a node in the roadmap. Otherwise, both are discarded. This method's performance is dependent on tuning the parameter d for each environment.
    3. Uniform OBPRM (UOBPRM) [25]: uses the uniform sampling framework to generate uniformly distributed configurations near C-obst surfaces by detecting when C-obst surfaces have been crossed. This is done by looking for validity changes between consecutive points along a defined line segment d. When a validity change occurs, the valid sample is retained

as a roadmap node. This method also needs some parameter tuning on d to get effective results.

- In Narrow Passages

  1. Bridge Test PRM [46]: This method was implemented to boost sampling density in narrow passage to improve the connectivity of roadmaps. In a bridge test they check for collision at three sampled configurations: the two endpoints and the midpoint of a short line segment. This method also suffers from parameter tuning which can greatly affect the performance and quality of the mappings produced.

  2. Toggle PRM [29,31]: performs a coordinated mapping of both C-free and C-obst. It retains witnesses from failed connection attempts in one space to augment the roadmap in the opposite space. [31] provides a novel classification of narrow passages that can be solved efficiently by this methodology. Toggle PRM is able to solve certain narrow passages, but it does not provide a guarantee on the solution quality.

  3. Medial Axis PRM(MAPRM) [71, 108]: Medial Axis PRM was proposed to generate configurations along the medial axis of C-free to increase the path clearance. The medial axis is a set of points that are equidistant to two or more obstacles and are guaranteed to have maximal clearance. The medial axis is a *strong deformation retraction* which makes a one-to-one mapping between every point in C-Space and the corresponding point on the medial axis and is thus a useful construction for motion planning. It makes use of fundamental primitives which are computation of penetration depth and clearance in C-Space to achieve this. However, MAPRM does not provide any guarantee regarding the distribution of samples along the

medial axis and it is expensive to produce samples.

4. Uniform MAPRM(UMAPRM) [109]: UMAPRM generates uniformly distributed samples along the medial axis by checking the closest obstacle changes between two neighboring configurations on the segment. The medial axis is crossed when there is a closest obstacle change. A binary search finds and retains the configuration on the medial axis.

2.3.2 Existing Connection Methods

Connection methods are primitives used in PRM to connect nodes via edges together while building a roadmap. The connection method defined within our context include a combination of a distance metric (used to calculate the distance between configurations), a neighbor finding approach (to identify pairs of "close/similar" configurations), and a local planner (to determine if a feasible path exists between two configurations). In this section we discuss these three primitives used during the connection phase for PRMs, i.e., neighbor finding methods, distance metrics and local planners.

- Distance Metrics

  A distance metric is a function δ that computes some "distance" between two configurations $a = \langle a_1, a_2, \ldots, a_d \rangle$ and $b = \langle b_1, b_2, \ldots, b_d \rangle$, i.e., $\delta(a, b) \rightarrow \mathbb{R}$, where d is the dimension of a configuration. A good distance metric for a PRM predicts how likely it is that a pair of nodes can be connected. In this research, we study the set of distance metrics commonly used in PRMs:

  1. Euclidean: The Euclidean distance metric gives equal weighting for all

dimensions:

$$\delta(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_d - b_d)^2}$$

2. The scaled Euclidean distance metric is a variant

$$\delta(a,b) = \sqrt{s(\text{pos\_mag})^2 + (1-s)(\text{ori\_mag})^2}$$

where pos_mag is the Euclidean distance of the positional dimensions, ori_mag is the Euclidean distance of orientational dimensions, and s is a weighting parameter. In the results presented here, we use $s = 0.5$ and refer to this as "Euclidean".

3. Minkowski: The Minkowski distance is the generalized form of the Euclidean distance which uses parameters r_1, r_2, and r_3 to specify the exponents and roots used:

$$\delta(a,b) = \sqrt[r_3]{(a_1 - b_1)^{r_1} + \cdots + (a_{d-1} - b_{d-1})^{r_2}}$$

The positional DOFs (usually the first 3 dimensions) use r_1 as a power factor and the orientation/joint DOFs use r_2 as a power factor – typically, $r_1 = r_2$ and $r_3 = 1/r_1$.

4. Manhattan: The Manhattan distance metric is the distance between two points if a grid path is followed:

$$\delta(a,b) = (|a_1 - b_1| + |a_2 - b_2| + \cdots + |a_d - b_d|)$$

5. RMSD: Root Mean Square Distance (RMSD) is measured as the average

distance between two objects X and Y. It records the level of similarity between the position and orientation of two objects:

$$\delta(X,Y) = \sqrt{\frac{(X_1 - Y_1)^2 + (X_2 - Y_2)^2 + \ldots + (X_d - Y_d)^2}{d}}$$

6. Swept Volume: Swept volume is the volume generated by the continuous motion (translation and/or rotation) of a geometric object through space. The swept volume distance is the volume swept by the robot while following the motion prescribed by the local planner. For an articulated linkage, this becomes the sum of the swept volumes of each of the links. This distance metric is expensive but more accurate for any local planner.

- Neighbor Finding Methods

1. K-Closest /R-Closest methods: These methods return the k closest neighbors to a node based on some distance metric, where k is normally some small constant, and can be done in logarithmic time. The advantage is that nodes are more likely to be connectable by the local planner because the volume of C-Space the connection occupies is smaller. Another approach is the r-closest method which returns all neighbors within a radius r of the node as determined by some distance metric. Here, the size of the neighbor set is not fixed but is dependent on the sampling density.

2. Randomized K-Closest variants: Two randomized variants of these methods are proposed in [78]: K-Closest,K-Rand and R-Closest,K-Rand. K-Closest,K-Rand randomly selects k neighbors from the k_2 closest nodes, where typically $k_2 = 3k$. R-Closest,K-Rand selects k random neighbors from those within a distance r. In some cases, these methods outperform

K-Closest as they introduce some useful randomness.

3. Reachablility Based Analysis [43]: This work describes the properties of these neighbor finding approaches and motivates research on connections based upon reachability analysis. However, these are expensive and should be limited to acquiring roadmap connectivity and/or seeking asymptotically shortest paths [56].

4. Metric Trees [104]: These data structures organize the nodes in a spatial hierarchical manner by iteratively dividing the set into two equal subsets resulting in a tree with $O(\log n)$ depth. However, as the dataset dimensionality increases, their performance decreases [72].

5. KD-trees [10]: These trees extend the binary tree into a D-dimensional data structure which provides a good model for problems with high dimensionality. However, a separate data structure needs to be stored and updated each time a node is added to the roadmap.

6. Approximate Neighbor Finding Methods

   These methods address the running time issue of exact neighbor finding approaches by instead returning a set of approximate K-Closest neighbors. These include spill trees [72], MPNN [110], and Distance-based Projection onto Euclidean Space [90]. These methods usually provide a bound on the approximation error.

- Local Planners

  A local planner (LP) connects two nodes with an edge based on defined closeness characteristics [5]. There are many local planners that can be used to connect two nodes a and b, and in this work we focus on two: *StraightLine and*

RotateAtS. We used the StraightLine local planner because apart from being commonly used, it is the fastest to compute and thus has less computation overhead. RotateAtS is useful as a comparison tool because it is an offshoot of the straightline local planner but does some rotation along the way. We also briefly describe other local planning methods called *TransformAtS and Toggle LP*.

1. StraightLine [5]: interpolates between two points in C-Space checking intermediate points on a straight line in C-Space. Although this local planner is simple and fast, it often fails in cluttered environments where nearest neighbors cannot be connected by a straight line due to the large swept volume.

2. RotateAtS [5]: reduces the swept volume by translating from a for some distance s toward b, changes all orientation DoFs, and translates again to get to b. The rotation allows the local planner some chance to get around obstacles making it more successful with samples that are close to obstacles.

3. TransformAtS is a modification of RotateAtS that changes all DOFs one by one when it gets to s.

4. Toggle LP [30]: a straight-line connection between the configurations a and b is attempted. If this fails then a third configuration n is generated that defines a triangle between a, n and b and a path will be searched within this triangle. This method extends local planning to a two dimensional plane of C space. Toggle LP can show a proof of disconnection (i.e., no valid path exists) but there is an added overhead of generating the third node which proves expensive as the complexity of the problem

increases.

2.3.3 Adaptive Learning Techniques for PRMs

In this section we discuss work related to this research including adaptive sampling, connection and overall planning for PRM methods.

2.3.3.1 Adaptive Learning during Sampling

Many techniques use machine learning to improve the performance of methods during the sampling phase of PRM roadmap construction. In this section we briefly highlight some of these methods.

1. Feature Sensitive Motion Planning [83] uses machine learning to help partition and characterize planning problems. Here, the planning space is subdivided in a recursive manner, then each region is classified and assigned an appropriate planning method. One main strength of this approach is its ability to map workspace/C-Space topologies for a particular planner. However, it is not able to adapt sampling methods over time.

2. HybridPRM [48] employs a reinforcement learning approach to select a node generation method that is expected to be the most effective at the current time in the planning process. However, these samplers are applied globally over the entire problem, and the features of the planning space, such as topology, are not used when deciding where to apply the selected method.

3. The Unsupervised Adaptive Strategy (UAS) [100] is similar to feature sensitive motion planning in the sense that it identifies regions and specifies the planner to the region. UAS also considers the topology of the space. In UAS, the K-means clustering method is used to partition the space using a training

roadmap and then hybrid PRM [48] is applied in each region. This method showed an improvement in speed and quality in the roadmaps generated, but does not consider all aspects of the planning process in particular, the node connection process.

4. Utility Guided Sampling [21, 22] uses information from previous experiences to guide sampling to more relevant areas of C-Space. Every exploration of C-Space provides information to the motion planner. They construct an approximate model of C-Space. Their model captures and maintains information from each configuration and predicts the state of unobserved configurations to reduce collision detection calls.

2.3.3.2 Adaptive Learning during Connection

This section focuses on the learning methods applied during the connection phase of PRM roadmap construction.

1. Adaptive Neighbor Connection (ANC): The work in [35] adaptively selects the appropriate connection method to use over time. It does so by maintaining a selection probability for each method based on previous performance. The main weakness of this approach is that it bases its decisions on the performance of connection methods over the entire environment in a global approach.

2.3.3.3 Adaptive Learning for Complete Planning

This section discusses the adaptive methods that focus on the overall approach, i.e., sampling and connection during PRM roadmap construction.

1. RESAMPL [95] uses local region information (e.g., entropy of neighboring samples) to make decisions about both how and where to sample, and which sam-

ples to connect together. This use of spatial information about the planning space enables RESAMPL to increase sampling in regions identified as narrow and decreases sampling in regions identified as free. These approaches do not consider the topology that is discovered within the explored space.

2. Learning from Experience [13] proposes a framework called Lightning that is able to learn from experience. Lightning consists of two modules that run in parallel: a planning from scratch module and a module that retrieves and repairs paths stored in the path library. Any path that is generated for a new query is checked by a library manager to decide how expensive the path is and how similar it is to previously generated paths. However, as the size of the library gets bigger, it becomes impractical to add new paths.

3. Apprenticeship Learning [1] uses inverse reinforcement learning and presents a refined algorithm that compares the trajectories with a more accurate metric and uses the algorithm in the context of apprenticeship learning. It solves problems within the context of motion planning by observing how expert agents behave, i.e., learn from demonstration.

4. Curiosity Driven PRM [40] utilizes reinforcement learning to enhance PRM planners for humanoids. To enhance time overhead of PRM as it plans (thinks) before executing actions, the authors created a modular behavioral environment (MoBeE) that implements a model-based reinforcement learner on planners. They assign probabilities to all possible actions from a given state and use them to identify interesting versus non interesting actions. This helps explore least visited areas, thus speeding up the planning stage of PRM. However this work is designed to work for humanoids and it is not a general PRM method.

Our work makes improvements on these different adaptive learning methods because we apply learning during both sampling and connection, identify neighborhoods(regions) in the heterogeneous environments dynamically, without the need for an explicit partition and our results show improvements due to this contribution.

3. LOCAL LEARNING*

In this Chapter, we discuss our framework based on the dynamic/implicit identification of local neighborhood (regions) within an environment and the utilization of the multi-arm bandit problem algorithms as briefly described in Chapter 2.

3.1 Local Learning

Our learning framework is a reward based learning method that utilizes the multi-armed bandit problem algorithms [11, 20]. This method combines the schematics of the reinforcement learning exploration and exploitation terminologies as described in Chapter 2.

We make modifications to these algorithms to include a local region for learning and the reward and cost in this region is recorded and reused to determine the next method suitable in that region when the next iteration begins. Our local learning approach focuses on the performance of the learning methods within a dynamically determined region. This dynamic region is determined based on a region specified by the node and its neighbors in which we are interested. Using reinforcement learning, each method is evaluated in terms of the cost and reward of previous attempts in that region. A method is rewarded as a function of every successful addition to total expected addition. The cost is expressed in terms of the number of collision attempts made by the local planner which directly affects the time taken to build the roadmap.

We focus on the performance during the different phases of PRM roadmap construc-

*The description of the method and some experimental results, tables and figures are reprinted with permission from "Improved roadmap connection via local learning for sampling based planners" by Ekenna C., Uwacu D., Thomas S., Amato N.M., 2015 IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 3227 - 3234 [38]

tion which are sampling and connection. The methods include all sampling methods earlier discussed in Section 2.3.1 and the connection methods described in Section 2.3.2. In our framework, there is a reward when a method successfully adds a node or an edge to the roadmap.

Algorithm 3 describes the learning algorithm. This is a general algorithm that can be used for sampling and connection. We initialize all the method's M to uniform probability and we update the probability using the UpdateProbability function in Algorithm 4. We determine the next method to perform an action based on the updated probabilities and call the PerformAction functions (Algorithm 6 and 7) which updates the cost and reward and also adds a configuration to the roadmap (sampling) or an edge (connection) based on required specifications. This algorithm takes inspiration from previous work on Hybrid PRM which looks into globally learning as described in [48]. In this work we apply this algorithms locally during both sampling and connection for PRM.

Algorithm 3 Learning(M)

1: Let P be a set of probabilities initialized to the uniform distribution, and M be a set of learning methods such that $\mid P \mid = \mid M \mid$.
2: P = UpdateProbability(n.method, n.reward, n.cost)
3: Select m based on P.
4: (reward,cost) = PerformAction(m)

The UpdateProbabilty function (Algorithm 4) is used to continually calculate and update the probabilities of the methods. This is important because this is where learning and keeping tabs on their performance is done. It shows the reinforcement learning calculations performed to obtain the probabilities determined for the

method's M.

Algorithm 4 UpdateProbability(m,$reward$,$cost$)

1: $w \leftarrow$ Update Weight using reward and m in Equation 3.1
2: $P_{nc} \leftarrow$ Calculate Probability without cost using w in Equation 3.2
3: $P_q \leftarrow$ Calculate Probability using P_{nc} , m and cost in Equation 3.3
4: **return** P_q

For each determined neighbor, we use Algorithm 5 to learn within each specified region with added input which includes the nearest neighbor finding method used to determine the local learning region. We store information about the method m in use, and reward and cost calculations for each m within the local learning region.

Algorithm 5 Local Learning(D, M, NF_{local})

1: Let D be data containing tuples (m, reward, cost), NF_{local} be a neighbor finding method and M be a set of learning methods such that $\mid P_q \mid = \mid M \mid$.
2: Let L be the learning region defined as the set of nearest neighbors to q given by NF_{local} in D.
3: **for** each $n \in L$ **do**
4: Learning (m)
5: **end for**
6: D $\leftarrow$ (m, reward, cost)

3.2 Local Learning during Sampling

Algorithm 6 describes the learning action performed during the sampling stage. We sample using the learned sampling method m from the set M, create a configuration q, and if is invalid, return the reward and cost as 0 and 1, respectively. Otherwise,

we connect the configuration q to the roadmap G. We return a reward of 1, if the current connected component *curr.count* is greater than or equal to the previous connected component count *prev.count* where *curr* = current and *prev* = previous. Otherwise, we make a calculation on how visible the configuration generated is.

Algorithm 6 Sampling
PerformAction(m)

1: Sample configuration q using m
2: **if** q is not valid **then**
3: **return** (0,1) where 1 is the sampling collision calls
4: **else**
5: Connect configuration to G.
6: **end if**
7: **if** $curr.count \geq prev.count$ **then**
8: $reward = 1$
9: **else**
10: $visibility = curr.succ/curr.att$
11: $reward = \epsilon^{-\gamma * visibility^2}$
12: **end if**
13: cost = # of collision calls after connection + # of collision call after sampling
14: **return** (reward, cost)

As defined in [84], a configuration q is visible to q' if there exists a path (e.g. a straight line) from q to q' that is entirely valid. In our analysis, a method that creates a configuration that increases the visibility of its connected component is more rewarded than one that adds a random configuration that over samples the connected component. We determine visibility as a function of current success recorded by the method divided by all the current attempts so far. The reward is thus an exponential function determined by the method's visibility. We determine the cost as the number of collision calls made after the connection has been made with the local planner including the collision call recorded after the configuration q has been sampled.

3.3 Local Learning during Connection

Algorithm 7 describes the learning application to the connection stage for PRM. We connect the configurations based on m from the set M and reward the methods based on the number of successful connections in ratio to the total connection attempts. We calculate the cost as the number of collision calls made after the connection has been made.

Algorithm 7 Connection
PerformAction(m)

1: Connect configuration q to G using m
2: reward = # of successful connections /Total connection attempts
3: cost = # of collision calls after connection
4: **return** (reward,cost)

First, methods are rewarded according to the number of their returned configurations that are successful. The reward is updated using the probability without the inclusion of cost because it should be independent of the accrued cost.

Let x_i be the reward for the method m_i that was selected. All other rewards for that time step are 0. To update the weights, we first take into account an adjusted reward that is not dependent on the cost accrued.

$$x_i^* = x_i/p_i^*, i = 1, 2, ...m. \tag{3.1}$$

After finding the updated reward, the weight is calculated as a function of the updated reward:

$$w_i(t+1) = w_i(t) \exp \frac{\gamma x_i^*}{m}, i = 1, 2, ..., m, \tag{3.2}$$

where x_i^* is the updated reward found by dividing the reward by the probability without the inclusion of cost. For the weights to adapt quickly, we use an exponential factor.

We then find the probability p_{nc}^* for each method m_i ignoring the cost:

$$p_{nc}^* = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{m} w_j(t)} + \gamma\frac{1}{m}, i = 1, 2, ..., m, \tag{3.3}$$

where $w_i(t)$ is the weight of m_i in step t, t is the number of connection attempts made by the planner, γ a fixed constant that represents the randomness of the method choice and m is the number of methods in the set. The probability calculation in Equation 3.3 is analogous to the total expected payoff measure for bandit algorithms. The aim is to maximize the expected payoff while minimizing the loss. We set *gamma* at 0.5 to ensure all methods have equal chances of being utilized. This formula computes the probability p_{nc}^* as a weighted sum of the relative weight of the m_i and the uniform distribution, which ensures that each method gets a chance to be selected.

We calculate a cost sensitive probability as a function of the cost insensitive one and the cost of connection attempts:

$$p_q = \frac{\frac{p_{nc}^*}{c_i}}{\sum_{j=1}^{m} \frac{p_j^*}{c_j}}, i = 1, 2, ..., m. \tag{3.4}$$

4. EXPERIMENTS IN ROBOT MOTION PLANNING*

In this Chapter, we discuss experiments performed using our local learning framework as discussed in Chapter 3. We perform experiments and show results when we apply our algorithm during the sampling and connection stage individually then we investigate the performance when we apply to both stages. These experiments were performed in a single and multi-query scenario. For the single query case, we examine the time to solve a query. For the multi-query case, we look at the roadmap coverage and connectivity (as measured by the number of queries solved). We begin this section by giving a brief overview about our previously published work where learning is applied globally because in our experiments, we compare the performance of local vs. global learning which further shows the benefit of our local approach.

4.1 Global Learning

Global learning [35] makes use of Algorithm 3 but continually rewards m from the set M based on its successful additions to the environment and the cost is expressed in terms of the number of collision detection attempts made. The reward and cost is updated based on the performance of the methods m on the entire environment, i.e., no dynamic determination of local neighborhood/regions.

If the environment is heterogeneous, the global learning performance would be hampered if the environment is not partitioned into regions because global learning would be forced to chose some neutral strategy or to vacillate between several strategies. In

*The description of the method and some experimental results, tables and figures are reprinted with permission from "Improved roadmap connection via local learning for sampling based planners" by Ekenna C., Uwacu D., Thomas S., Amato N.M., 2015 IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 3227 - 3234 [38] ©2015 IEEE.

such a situation, it is desirable to subdivide the problem into homogeneous regions and apply global learning in each one.

To achieve comparable results we employed an explicit partition method taking the cue from the previously implemented spatial subdivision method discussed in [18,85, 95, 112].

4.2 Experimental Setup

We look at a variety of input problems including 2D and 3D environments (see Figure 4.1) in industrial settings, narrow regions and highly heterogeneous environments.

- **2D-Heterogeneous, rod-like rigid robot.** (Figure 4.1(a)) A long rectangular rigid robot in a heterogeneous environment containing 8 different rooms of different types including cluttered, free, and blocked regions. The start is at the bottom left, and the goal is located at the top left. The robot must traverse each room to solve the query.

- **3D-Heterogeneous, spherical rigid robot.** (Figure 4.1(b)) A spherical robot must traverse 4 rooms separated by walls . These rooms include very narrow, cluttered, and maze regions. We allow roadmap construction to take 1200 seconds and provide 8 samples spread uniformly for querying. We then measure performance as the percentage of possible queries solvable from these samples by the roadmap.

- **6 DOF Manipulator arm.** (Figure 4.1(c)) A 6 degree-of-freedom manipulator arm with initial and final configurations inside a narrow passage where the robot has to pull out an object successfully from the bottom right window and place it in the bottom right window. This space between the robot and

the wall is made so narrow so that the robot would have to rotate 360 degree in the opposite direction to get to the other window.

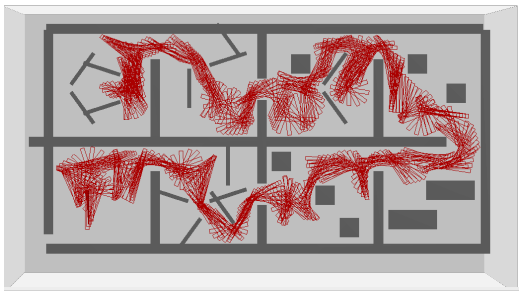
- **8 DOF KukayouBot [60].** Figure 4.1(d) An 8 DOF robot in an environment with four different rooms. Its base has 5 DOFs that allow it to move forward, backward and rotate, and its arm has 3 DOFs. The robot moves through different rooms with narrow passages and arrives at a destination where it performs an action (grasps or puts an object down).

The sampling methods that we compare to are Uniform sampling [58], OBPRM [8], Gauss [17], Bridge Test [46],UOBPRM [25], global learning for sampling (GLS), and our local learning applied to sampling (LLS). For connection methods, we use K-Closest, K-Closest,K-Rand, R-Closest,K-Rand, with $k = 10$ and $k_2 = 3k$ as determined in [79], and r as the average pairwise distance among a sample set of configurations. global learning for connection (GLC) and local learning for connection (LLC).

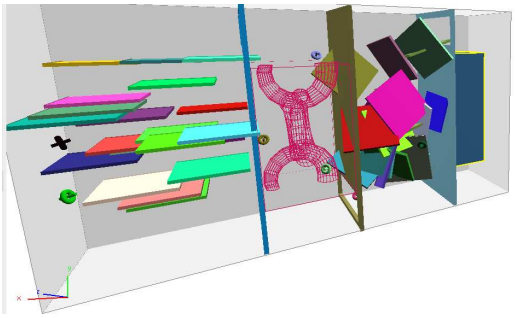
We use the StraightLine local planner and the RotateATS local planner [5] with $s = 0.5$. RotateATS attempts to find a path by translating from configuration q_A to q_B the portion of the distance denoted by s, rotating about its axis, and then translating the remaining way to q_B.

4.2.1 Single Query Results

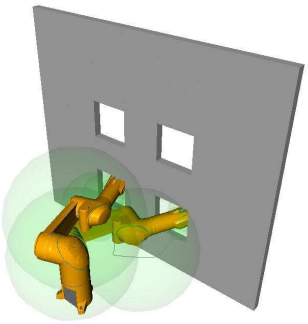
Here we show the performance in single query scenarios. Roadmaps are incrementally constructed until the query is solved. We measure performance as the time to solve the query and compare results using the various connection methods, global learning methods and local learning methods.



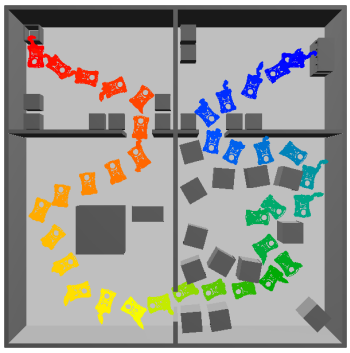
(a) 2D Mix



(b) 3D Maze Heterogeneous



(c) 6 DOF Manipulator Arm with a Fixed Base



(d) 8 DOF KUKA youBot model with a mobile base

Figure 4.1: Environments studied.

4.2.1.1 Learning during Sampling

In this section we apply learning during the sampling phase only (LLS) and compare with individual methods and GLS.

- 2D Mix Environment (Figure 4.1(a))

  Figure 4.2 shows the time to solve the query for each sampling method studied in the context of StraightLine local planning (red bars and y axis on the left) and RotateATS local planning (green bars and y axis on the right). OBPRM performs best using the StraightLine local planner and Gaussian using the RotateATS local planner. Using GLS and LLS, we see that they perform

similarly and second best to the best performing methods for StraightLine and comparable for RotateATS with the best performing method.

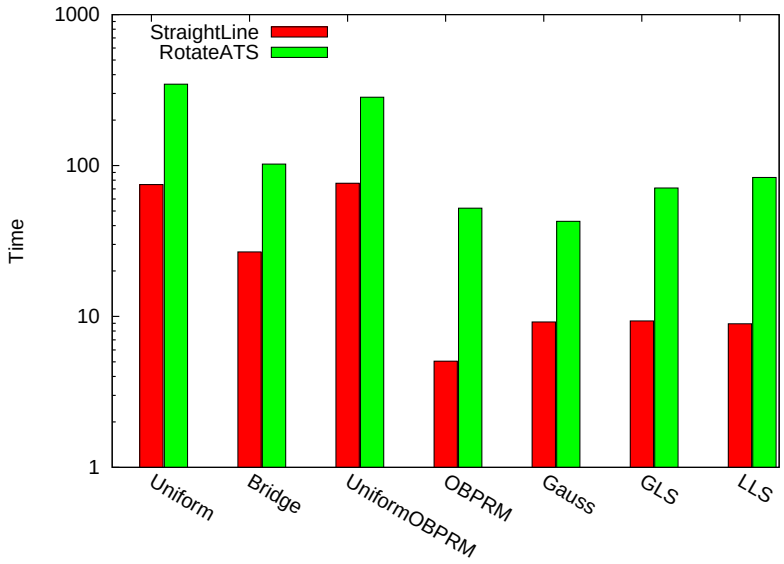


Figure 4.2: Query time for 2D Heterogeneous for Different Sampling Methods

- 6 DOF Manipulator Arm with a Fixed Base

  We provide results for the time, collision call, number of nodes and number of edges produced and the edge to node ratio (in node degree).

  The results in Table 4.1 show that for individual methods, OBPRM uses the least amount of time while Bridge test uses the smallest number of nodes but the time taken to solve the query indicates that these nodes are expensive due to the time Bridge Test needed to solve the query. Results with GLS show some improvement in the time second to OBPRM. However, results using LLS show an improvement in performance both in regards to the time taken to solve the

query and a reduced number of connected components. LLS performs better than the best individual method.

Table 4.1: Each method constructs a roadmap until the query is solved. GLS and LLS is comprised of the other 4 sampling methods. All results are averaged over 10 runs. CC is number of connected components present. Boldface entries indicate the most desirable (e.g., shortest running time, Nodes, Edges etc.).

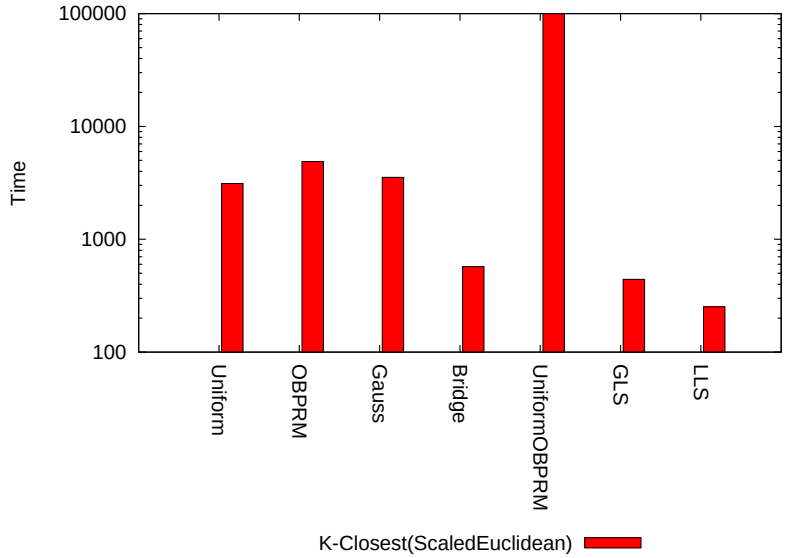
Environment	Method	Nodes	Edges	Edge/Node	Total Time (s)	CC
Stationary 6 degree-of-freedom manipulator arm, $k = 10$	BridgeTest	**3833**	54004	14.1	1373	338
	Gauss	18739	358575	19.1	644	364
	Uniform-PRM	58117	1150213	19.9	6048	1711
	OBPRM	9708	122109	12.5	344	42
	GLS	10762	146442	13.6	1078	47
	LLS	7022	100576	14.3	**339**	**36**

- 8 DOF KUKA youBot model with a mobile base

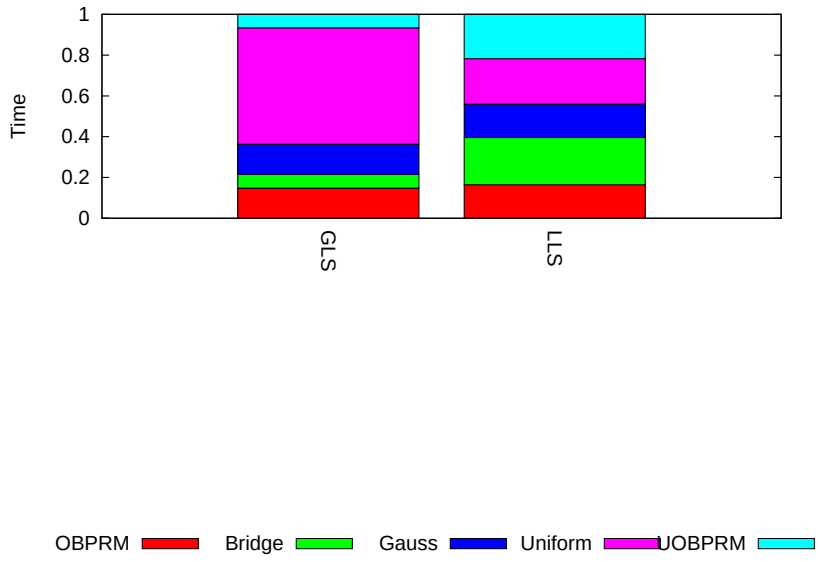
  We perform single query experiments and record the time needed to solve the query.

  Figure 4.3(a) shows the time needed to solve the query in the KukayouBot environment using the different sampling methods listed. Here we determine how each of the sampling methods performs including evaluating both global and local learning during the sampling phase.

  In Figure 4.3(a) we see that the LLS and GLS are better than all the individual methods. LLS performs better than GLS in this experiment which indicates that learning is important during the sampling phase. The Bridge Test performs better in terms of time to solve the query than the other sampling methods where learning is not applied.



(a) Query Time of Sampling Methods



(b) Percentage Frequency of Used Sampling Methods

Figure 4.3: Query time and Frequency of Usage for Different Sampling Methods

Figure 4.3(b) shows the frequency of usage of the different sampling methods in GLS and LLS. We see that GLS in most cases learns to use Uniform/Basic PRM sampling which is the simplest algorithm and thus would record a smaller cost

which GLS leverages on. However it does not learn the Bridge Test sampling method which from our results (see Figure 4.3(a)) is the better performing individual sampling method. LLS utilizes all the available sampling methods efficiently and it records the smallest time needed to solve the query.

4.2.1.2 Learning during Connection

In this section, we investigate the performance of the local learning method during the connection stage (LLC) and compare performance with individual methods.

- 2D Mix Environment

  Figure 4.4 shows the time to solve the query for each method studied in the context of StraightLine local planning (red bars and y axis on the left) and RotateATS local planning (green bars and y axis on the right). For StraightLine, R-Closest,K-Rand solves the query in the least time followed by LLC. LpSwept performs the worst because of its high computational cost. LpSwept is a more accurate method but due to the area covered during a sweep of the robot, such accuracy is not needed for StraightLine local planning. We also see that there is more time overhead needed to solve the query using RRT in comparison to both GLC and LLC. Note that LLC does better than GLC because the environment is heterogeneous, and there are no explicit subdivisions. Thus, local learning is needed.

  With RotateAtS, we see a change in performance of the individual connection methods. Specifically, we see that LpSwept performs better here than before. ScaledEuclidean performs worse than LpSwept because it does not predict well the feasibility of RotateATS as it over-penalizes candidates with large rotation differences. LLC was able to take advantage of this change in performance

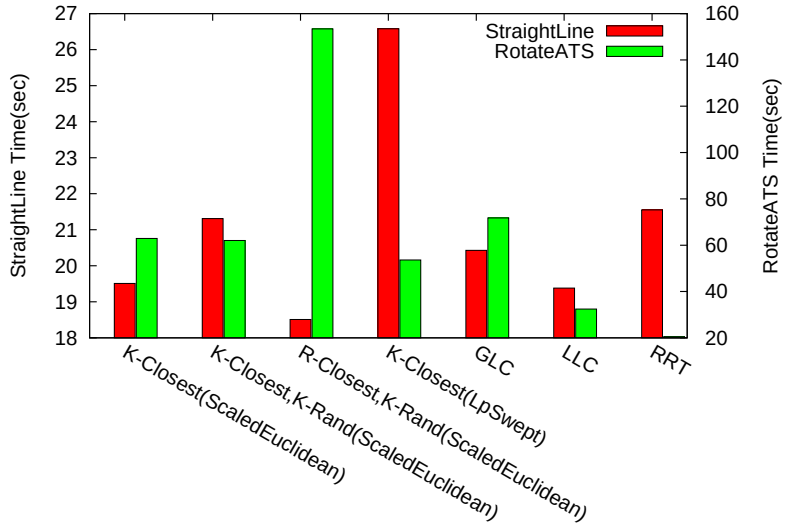


Figure 4.4: Running time in the 2D Mix Heterogeneous environment using different local planners averaged over 10 runs.

and clearly outperforms all the other methods. Again, LLC outperforms GLC because of the issue of heterogeneity.

- 8 DOF KUKA youBot model with a mobile base

  The results in Figure 4.3(a) help us select the sampling method (Bridge Test) to utilize for this experiment. Figure 4.5(a) shows time needed to solve the query using the Bridge Test as a sampling method and the different connection methods including GLC and LLC. We perform this experiment to determine if applying learning during the connection stage is beneficial.

  From the plots, we see that LLC outperforms all the other methods. It outperforms the other methods by a magnitude of 10 as is the case with the LpSwept method. This result indicates that learning is indeed important during the connection phase.

Figure 4.5(b) shows the performance frequency of usage of the connector methods for learning employed during the connection stage. Here we see that GLS learns LpSwept which is not one of the better connection methods. GLC's performance as earlier discussed in [35] is a result of the need to partition the environment to get good results which we did not do in these experiments. LLC however, utilizes the methods more which is an important feature of the local learning approach, i.e., their ability to utilize resources in a more intelligent way, which in this case is being able to use the methods available as the need arises.

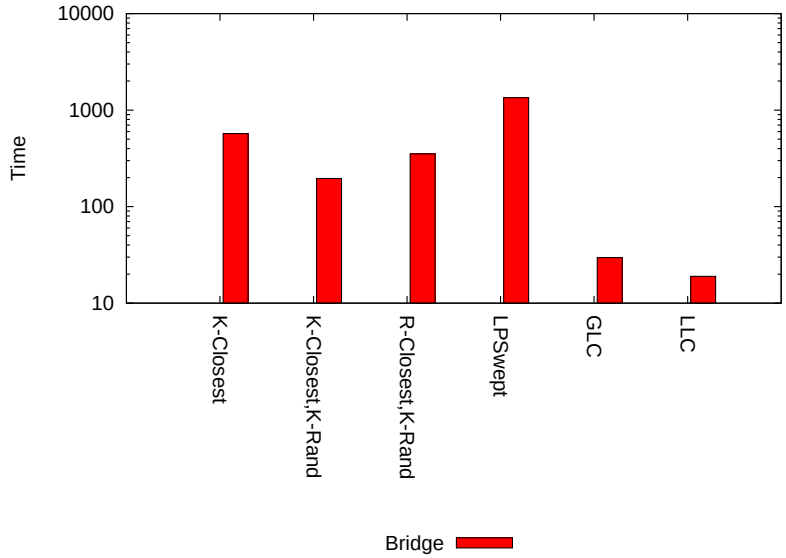
4.2.1.3 Learning in Both Phases

- 8 DOF KUKA youBot model with a mobile base

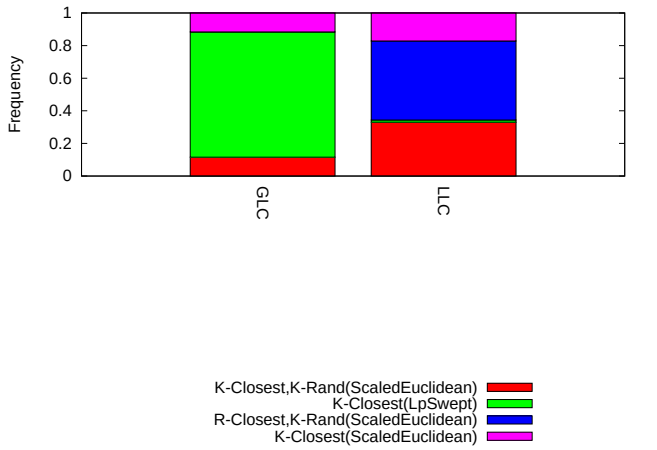
  Figure 4.6(a) shows the time needed to solve the query when learning (global and local) is applied to both the sampling and connection phase. Our results show that applying local learning during sampling and global learning to connection solves the query in the shortest time.

We see that applying local learning to sampling and global learning to connection is the best performing combination, followed closely by a global sampling and then local connection approach. Figure 4.6(b) shows the frequency of usage during learning both for sampling and connection and we see that the methods utilize all available methods in it's list better than other methods.

GLS in most cases learns to use Uniform/Basic PRM sampling which is the simplest algorithm and thus would record a smaller cost which the global method would leverage on. However, global connection learning in most cases picks the LpSwept method which is a method that spans the volume of the space when identifying



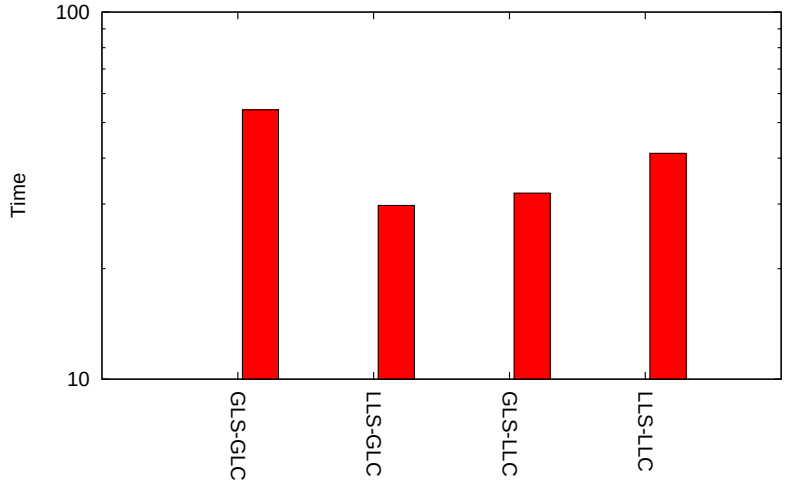
(a) Query Time of Connection Methods



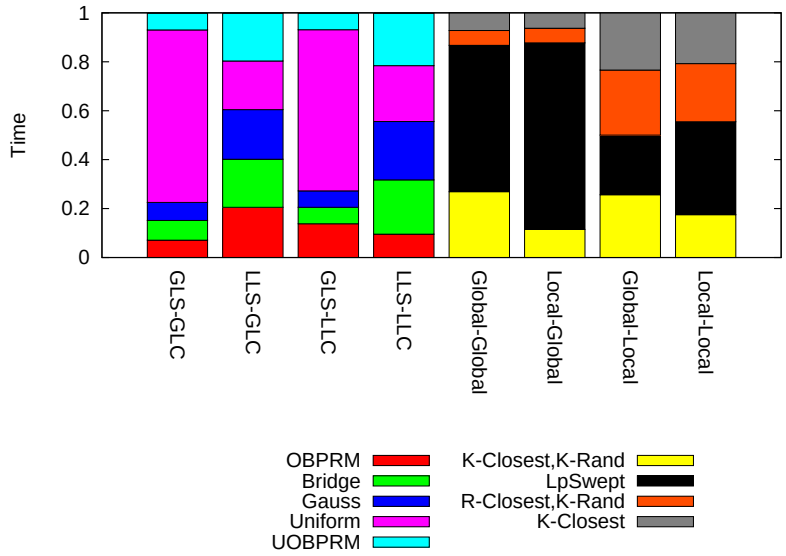
(b) Percentage Frequency of Used Connection Methods

Figure 4.5: Query time and Frequency of Usage for Different Connection Methods using Bridge Test Sampling Method

neighbors which is more effective but tends to be more expensive.



(a) Query Time with Learning Applied to both Sampling and Connection



(b) Percentage Frequency of Usage during the Sampling and Connection Phase

Figure 4.6: Query Time and Frequency of Usage during both phases of PRM construction

4.2.2 Multi-Query Experiments

In this section, we show the performance of the various methods in multi-query scenarios. Roadmaps are constructed within a fixed amount of time. We then count how many queries from a set of random samples are solvable by the resulting roadmap. We also look at the percentage of roadmap nodes contained in the largest connected component. This gives us indicators for coverage and connectivity. We perform multi-query experiment in the 3D Maze Environment and show behaviors of learning in the sampling and connection stages and when appied to both.

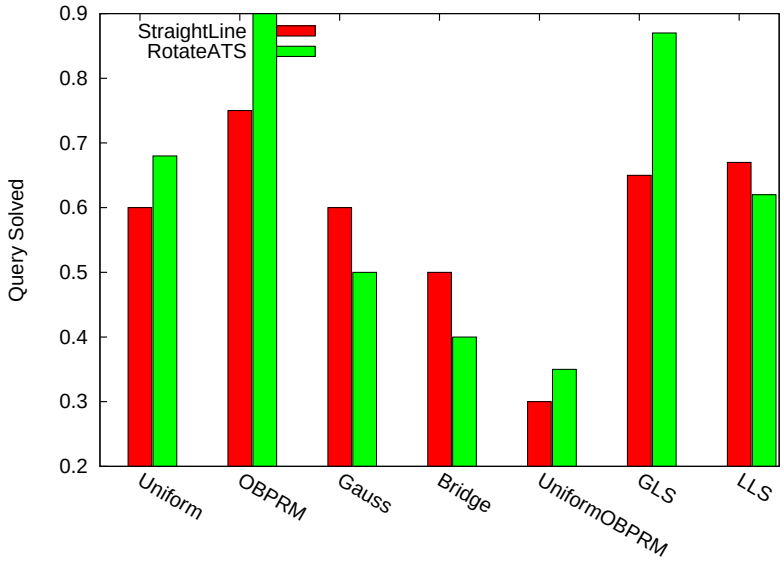
4.2.2.1 Learning in Sampling

Figure 4.7(a) shows that the sampling method using OBPRM solves more queries than the other methods. We see a higher percentage of nodes in its largest connected component as shown Figure 4.7(b) using the RotateAtS local planner and Straight-Line local planner when compared with the individual methods. Using the GLS, we see an improvement in performance for the StraightLine planner and a comparable method to OBPRM using the LLS the RotateAtS planner.

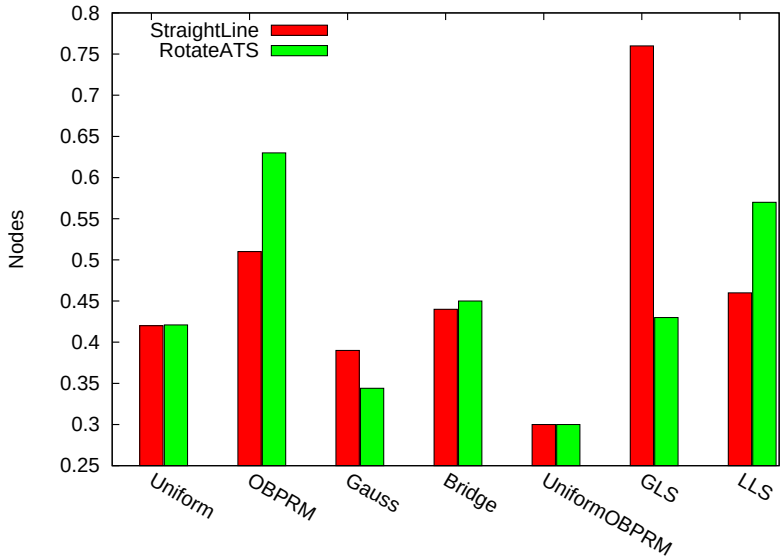
4.2.2.2 Learning in Connection

Figure 4.8(a) (bars in red), shows the percentage of queries solved and Figure 4.8(b) (bars in red) the percentage of nodes in the largest connected component for each method using the StraightLine local planner. LLC solves more queries than the other methods by almost a factor of 2. We see a comparably higher percentage of nodes in its largest connected component. LLC outperforms global connection learning due to the absence of explicit partitioning.

Figure 4.8(a) (bars in green) shows the results using the RotateATS local planner.



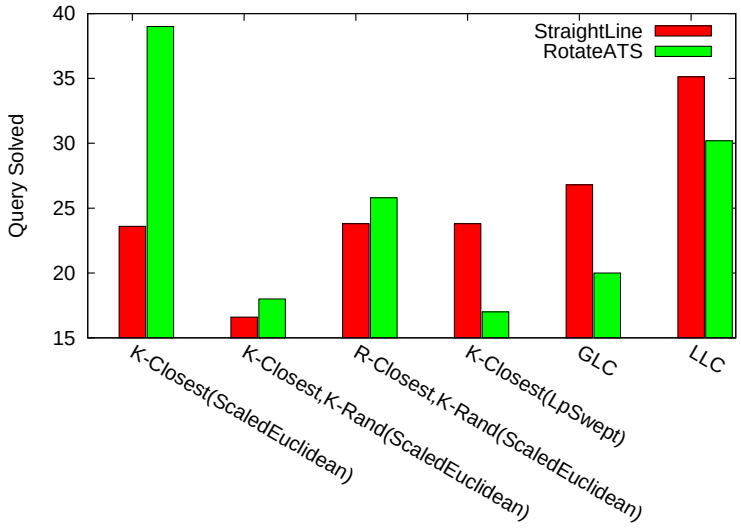
(a) Percentage of Queries Solved



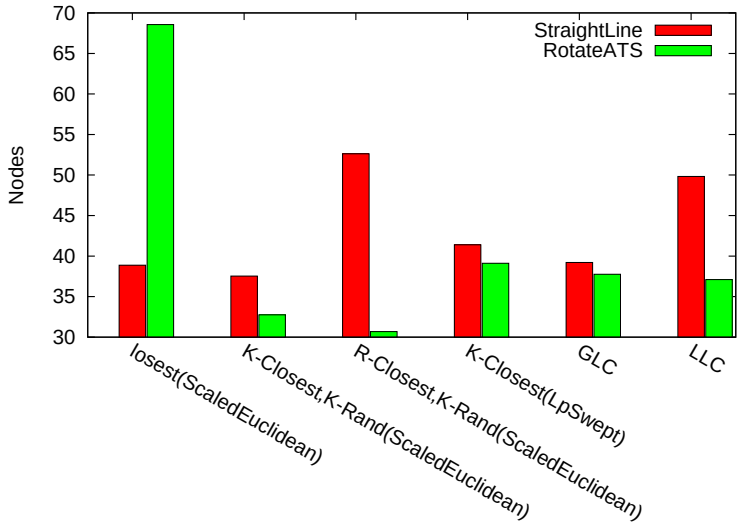
(b) Percentage of Nodes in the Largest Connected Component

Figure 4.7: Learning in Sampling for the 3D Maze Heterogeneous environment using different local planners averaged over 10 runs.

LLC is the second best in terms percentage of queries solved. Individual connection method performance has changed with the different local planners, but LLC still outperforms global connection learning.



(a) Percentage of Queries Solved



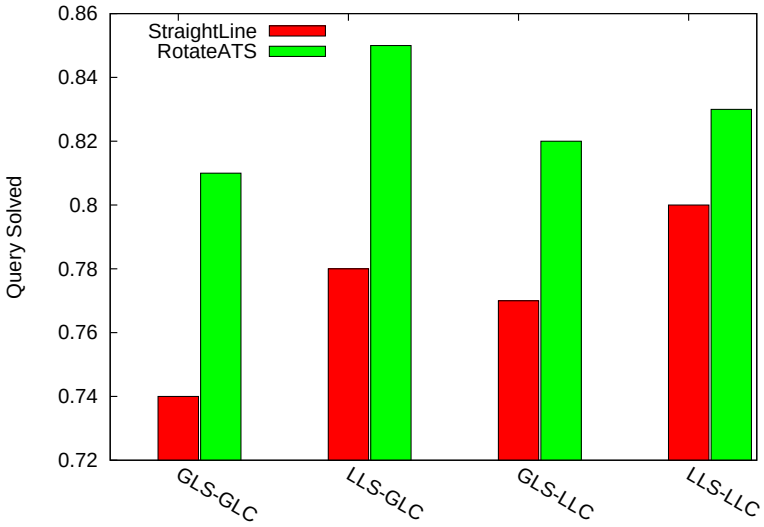
(b) Percentage of Nodes in the Largest Connected Component

Figure 4.8: Learning in Connection for the 3D Maze Heterogeneous environment using different local planners averaged over 10 runs.

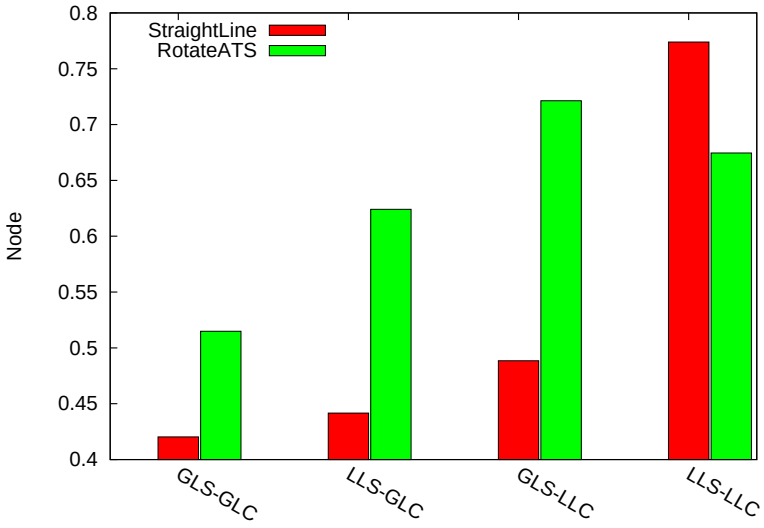
4.2.2.3 Learning in Both Phases

In this experiment, we look to see what happens when learning is applied to both phases of PRM construction during a multi-query scenario. Figure 4.9(a) shows the

percentage of queries solved and Figure 4.9(b) the percentage of nodes in the largest connected component. The results show that LLS-LLC performs best using both the RotateATS local planner and the StraightLine local planner.



(a) Percentage of Queries Solved



(b) Percentage of Nodes in the Largest Connected Component

Figure 4.9: Learning in Both Phases for the 3D Maze Heterogeneous environment using different local planners averaged over 10 runs.

5. LOCAL LEARNING AND PROTEIN FOLDING*

Modeling the protein folding process is crucial in understanding not only how proteins fold and function, but also how they misfold triggering many devastating diseases (e.g., Mad Cow and Alzheimer's [23]). Knowledge of the stability, folding, kinetics, and detailed mechanics of the folding process may help provide insight into how and why the protein misfolds. Since the process is difficult to experimentally observe, computational methods are critical.

Traditional computational approaches for generating folding trajectories such as molecular dynamics [69], Monte Carlo methods [27], and simulated annealing [68] provide a single, detailed, high-quality folding pathway at a large computational expense. As such, they cannot be practically used to study global properties of the folding landscape or to produce multiple folding pathways. The use of massive computational resources, such as tens of thousands of PCs in the Folding@Home project [12, 64] have helped improve the time overhead involved but still are unable to handle very large proteins. Statistical mechanical models have been applied to compute statistics related to the folding landscape [19, 88]. While computationally more efficient, they do not produce individual pathway trajectories and are limited to studying global averages of the folding landscape.

Robotics-based motion planning techniques, including the Probabilistic Roadmap Method (PRM), have been successfully applied to protein folding [6, 9, 26]. They construct a roadmap, or model, of the folding landscape by sampling conformations

*The description of the method and some experimental results, tables and figures are reprinted with permission from "Adaptive local learning in sampling based motion planning for protein folding" by Ekenna C., Thomas S., Amato N.M., 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 61-68 [36] ©2015 IEEE.

and connecting neighboring ones together with feasible transitions using a simple local planner. They can generate multiple folding pathways efficiently (e.g., a few hours on a desktop PC) enabling the study of both individual folding trajectories and global landscape properties.

We apply our local learning framework to study the protein folding process and we apply this framework to the sampling and connection stages of PRM construction used for simulating protein folding. We examine the performance of our method on 23 proteins of varying secondary structure makeup with 52–114 residues. We examine both the time to build roadmaps and the resulting trajectory quality. We also compare secondary structure formation order as exhibited in the pathways our roadmaps and experimentally determined where available. Our results confirm that learning is necessary, as no individual method is the best choice for all proteins. We also show that local learning generates better quality trajectories and is comparable in time to the best individual method for each individual input.

5.1 Related Work and Preliminaries

In this section we describe some preliminaries and related work including experimental protein dynamics, the protein model used, PRMs for protein folding and the distance metrics used specifically for protein folding. We then discuss existing machine learning techniques for PRMs and for protein motion and analysis.

5.1.1 Experimental Protein Dynamics

There have been several advances in experimental techniques to study protein dynamics and motion including circular dichroism, fluorescence experiments, hydrogen exchange and pulse labeling, NMR spectroscopy, and time-resolved X-ray crystallography. We briefly discuss each in turn.

- Circular dichroism (CD) is a spectroscopic technique used to investigate the structure and conformational changes of proteins [73]. By informing on binding and folding properties, CD provides information about the protein's biological functions. The CD signal occurs when chromophores in an asymmetrical environment interact with polarized light. In the case of proteins, the main chromophores are the peptide bonds as they absorb polarized light in the far-UV wavelength region (i.e., below 240 nm).

- Fluorescence spectroscopy analyzes the emission of fluorophores in the protein as the protein undergoes conformational change [87], such as during folding or upon binding. These fluorophores act as indicators of the state of the local environment, e.g., how structured the portion of the protein is near the fluorophore. As almost all proteins have natural fluorophores (i.e., tyrosine and tryptophan residues), fluorescence spectroscopy has broad applicability.

- Hydrogen exchange mass spectrometry and pulse labeling can investigate protein folding by identifying which parts of the structure are most exposed or most protected [77]. From this data, one can infer which portions of the protein fold first and which are last to form, up to the millisecond timescale.

- NMR spectroscopy, another experimental tool often used to study protein dynamics, is a technique used to determine a compound's unique structure. It identifies the carbon-hydrogen framework of an organic compound and has been used to study side-chain motion and backbone motion [45]. See [96] for a review of current techniques.

- X-ray crystallography obtains a three dimensional molecular structure from a crystal [97]. A purified sample at high concentration is crystallized and the

resulting crystals are exposed to an x-ray beam. This produces a pattern of diffraction spots. The intensities of these spots can be used to determine the structure factors from which an electron density map can be calculated.

While experimental methods can probe some fine-grained details of protein motion, they are time intensive and limit the time scales they can access. In addition, experimental methods may not be able to be applied to all proteins, e.g., some proteins naturally precipitate out and cannot be analyzed. Simulations, instead, affords the opportunity to study such proteins and others much faster (hours vs. days) with computational resources which will potentially save both time and money.

5.1.2 Protein Model

Proteins are sequences of amino acids, or residues. We model the protein as a linkage where only the ϕ and ψ torsional angles are flexible, a standard modeling assumption [76]. A potential energy function models the many interactions that affect the protein's behavior [69]. This function helps quantify how energetically feasible a given conformation is.

In this work, we employ a coarse-grained potential function [6] which helps define some characteristics of our modeling framework; If the atoms are too close to each other (less than 2.4Å in sampling and 1.0Å in connecting), the conformation is unfeasible; otherwise, the energy is calculated by:

$$U_{tot} = \sum_{constraints} K_d\{[(d_i - d_0)^2 + d_c^2]^{1/2} - d_c\} + E_{hp} \tag{5.1}$$

where K_d is 100 kJ/mol, d_i is the length on the ith constraint, E_{hp} is the hydrophobic interaction, and $d_0 = d_c = 2$Å as in [69]. The coarse grain model has been shown

to produce qualitatively similar results as all-atoms models but faster in terms of time [98].

5.1.3 PRM for Protein Folding

The Probabilistic Roadmap Method (PRM) [58] is a robotics motion planning algorithm that first randomly samples robot (or protein) conformations, retains valid ones, and then connects neighboring samples together with feasible motions (or transitions). To apply PRMs to proteins, the robot is replaced with a protein model and collision detection validity computations are replaced with potential energy calculations [4, 6, 9, 26].

5.1.3.1 Sampling

Protein conformations, or samples, are randomly generated with bias around the native state, the functional and most energetically stable state. Samples are iteratively perturbed, starting from the native state, and retained if energetically feasible by the following probability:

$$P(q) = \begin{cases} 1 & \text{if } E(q) < E_{min} \\ \dfrac{E_{max} - E(q)}{E_{max} - E_{min}} & \text{if } E_{min} < E(q) \leq E_{max} \\ 0 & \text{if } E(q) > E_{max} \end{cases} \tag{5.2}$$

where E_{min} is the energy of the open chain and E_{max} is $2E_{min}$. We use rigidity analysis to focus perturbations on flexible portions as detailed in [103]. We perturb flexible torsional angles with a high probability, P_{flex}, and rigid torsional angles with a low probability, P_{rigid}. Perturbing rigid torsional angles ensures coverage of the landscape. This method provides a denser distribution of samples near the native

conformation and focuses sampling on currently flexible regions.

5.1.3.2 Connection

Once a set of samples is created, they must be connected together with feasible transitions to form a roadmap, or model, of the folding landscape. Connecting all possible pairs of samples is computationally unfeasible, and it has been shown that only connecting the K-Closest neighbors results in a roadmap of comparable quality [79].

Given a pair of samples, we compute a transition between them by a straight-line interpolation of the ϕ and ψ torsional angles. Straight-line local planning involves the fewest number of intermediates to check for validity and has been shown to be a sufficient measure of transition probability; i.e., it can accurately predict secondary structure formation order [6, 98]. We assign an edge weight to reflect the energetic feasibility of the transition as $\sum_{i=0}^{n-1} -log(P_i)$ where P_i is the probability to transit from intermediate conformation c_i to c_{i+1} based on their energy difference $\Delta E_i = E(c_{i+1}) - E(c_i)$:

$$P_i = \begin{cases} e^{\frac{-\Delta E_i}{kT}} & \text{if } \Delta E_i > 0 \\ 1 & \text{if } \Delta E_i \leq 0 \end{cases} \tag{5.3}$$

where k is the Boltzmann constant and T is the temperature. This allows the most energetically feasible paths to be extracted by standard shortest path algorithms.

5.1.3.3 Validation by Secondary Structure Formation Order

Proteins are composed of secondary structure elements (i.e., α-helices and β-strands). Experimental methods, such as hydrogen exchange mass spectrometry and pulse labeling, can investigate protein folding by identifying which parts of the structure

are most exposed or most protected [107]. From this data, one can infer the secondary structure formation order.

In [6, 76, 98], we compared the secondary structure formation order of folding pathways extracted from our maps to experimental results [70] by clustering paths together if they have the same formation ordering. We return a stable roadmap when the distribution of secondary structure formation orderings along the folding pathways in the graph stabilizes, i.e., the percentage of pathways following a given ordering does not vary between successive graphs by more than 30%. As our roadmaps contain multiple pathways, we estimate the probability of a particular secondary structure formation order occurring by the percentage of roadmap pathways that contain that particular formation order. The roadmap corroborates experimental data when the dominant formation order (i.e., the one with the greatest percentage) is in agreement.

5.1.3.4 Distance Metrics

The distance metric plays an important role in determining the best connections to attempt. It is a function δ that computes some "distance" between two conformations $a = \langle a_1, a_2, \ldots, a_d \rangle$ and $b = \langle b_1, b_2, \ldots, b_d \rangle$, i.e., $\delta(a, b) \rightarrow \mathbb{R}$, where d is the dimension of a conformation. Here, $a_1 \ldots$ and $b_1 \ldots$ are the ϕ and ψ torsional angles for each protein conformation. A good distance metric generally predicts how likely it is that a pair of nodes can be successfully connected. Their success is dependent on the nature of the problem studied. We use the following set of distance metrics commonly used when applying motion planning to protein motion:

- Euclidean Distance Metric

  The Euclidean distance metric captures the amount of physical movement

(around the torsional angles) that conformation a would undertake to move to conformation b. This distance is computed by measuring the difference in the ϕ and ψ angle pairs of the two conformations:

$$\delta_{\text{Eucl}}(a,b) = \sqrt{\frac{(\phi_1^a - \phi_1^b)^2 + (\psi_1^a - \psi_1^b)^2 + ... + (\phi_n^a - \phi_n^b)^2 + (\psi_n^a - \psi_n^b)^2}{2n}}. \quad (5.4)$$

- Cluster Rigidity Distance Metric

  Rigidity analysis [50] computes which parts of a structure are rigid and flexible based on the constraints present. It may be used to define a rigidity map r, which marks residue pairs i, j if they are in the same rigid cluster. Rigidity maps provide a convenient way to define a rigidity distance metric, between two conformations a and b where n is the number of residues:

$$\delta_{\text{Rig}}(a,b) = \sum_{0 \le i < j \le 2n} (r_a(i,j) \neq r_b(i,j)). \quad (5.5)$$

  More details may be found in [103].

- Root Mean Square Distance Metric

  The protein model has 6 atoms for each amino acid. Thus, a protein with n amino acids will have $6n$ atoms. Denoting the coordinates of these atoms as x_1 to x_{6n}, the root mean square distance (RMSD) between conformations a and b is:

$$\delta_{\text{RMSD}}(a,b) = \sqrt{\frac{(x_1^a - x_1^b)^2 + (x_2^a - x_2^b)^2 + ... + (x_{6n}^a - x_{6n}^b)^2}{6n}}. \quad (5.6)$$

  Least RMSD (lRMSD) is the minimum RMSD over all rigid body superpositions of a and b.

5.1.4 Machine Learning for Protein Analysis and Motion

Machine learning algorithms have been employed to predict protein folds, estimate folding rates, and study folding motions. We highlight a few relevant techniques here.

- Protein Fold Recognition

  Protein fold recognition involves identifying the correct structural fold from among a set of known template protein structures for a given protein sequence. Fold recognition is essential for template-based protein structure modeling. The fold recognition problem is defined as a binary classification problem of predicting whether or not the unknown fold of the input protein is similar to an already known template from a protein structure library.

  RF-Fold uses random forests, a highly scalable classification method, to recognize protein folds [53]. A random forest is composed of many decision trees that are each trained on datasets of target-template protein pairs. RF-Fold recognition rate is comparable to the best performance in fold recognition at the family, superfamily, and fold levels.

  DN-Fold is another fold recognition technique, but it uses a deep learning neural network as a basis for learning [54]. A deep learning network has many more layers than a typical neural network. In addition, they may be trained through unsupervised learning. Deep learning was applied to fold prediction by restating the problem as predicting if a given target-template pair belonged to the same fold. They showed that DN-Fold achieved comparable performance over a wide variety of methods at all three fold levels.

- Folding Rate Prediction

In addition to predicting the fold of a protein, it is useful to estimate its folding rate. This is important when studying properties such as stability and classifying kinetics. Characteristics of the protein structure, such as contact order and total contact distance, affect the folding rate. However, the precise relationship between these characteristics and the rate are unknown. A back-propagation neural network was used to quantify this relationship [113]. Their results showed that correlations exist between these properties and the folding rate with relative errors for predicted results lower than competing methods.

- Simulating Protein Motion Trajectory

  Machine learning has also been applied to studying protein folding trajectories. In [41] they use unsupervised learning to cluster similar states and basins present in the folding landscape. They then use this clustering to construct an exploration bias to speed up molecular dynamics simulations. Specifically, the exploration bias guides the next basin to jump to in the simulation while ensuring that the entire conformation space is explored. They provide simulation results for an alanine trajectory.

5.2 Learning Framework for Protein Folding

We use the same learning framework algorithms described in Chapter 3 and make modifications to the reward and cost based on the potential energy calculations. Potential energy computations take up a large portion of the total computation time and thus are a good measure of the reward and cost. Here, we calculate the cost as the number of potential energy calls incurred by the connection method. This would be used both during learning for sampling and connection. We make this change from what we previously used in our robotics experiment which involved a calculation of

the number of successful connections made to what was expected to now a function based on the potential energy. The energy function gives a clear indication of valid configurations and as seen in Equation 5.7 we make use of the normalized function to bring all our variables in proportion to one another. We make modifications to the reward as described below.

Let x_i be the reward for the cm_i that was selected:

$$x_i = \alpha + (1 - \alpha)(1 - \frac{y_i(t) - miny_i(t)}{maxy_i(t) - miny_i(t)}) \tag{5.7}$$

where $y_i(t)$ = current edge weight, $miny_i(t)$ = minimum edge weight recorded during the current step, $maxy_i(t)$ = maximum edge weight recorded during the current step, and α = a constant value used to normalize the reward. All other rewards for that time step are 0. The reward is thus a function of the edge quality (weight) and the local planner's success.

5.3 Experiments

We study 23 proteins (see Table 5.1) with 52–114 residues. This set contains α, β, and mixed proteins that were also studied by [34] and many have experimentally determined secondary structure formation orders [74]. The protein structures were obtained from the Protein Data Bank [14]. We perform experiments by applying our learning framework during sampling and connection phases of PRM roadmap construction.

Metrics are computed as follows:

- *Secondary Structure Formation Order:* We compare, when available, the secondary structure formation order predicted by each method to experimental

Table 5.1: Proteins studied.

Protein Name	PDB ID	Length	Secondary Structure
Rubredoxin	1RDV	52	$2\alpha + 2\beta$
Ferredoxin	1FCA	55	$2\alpha + 2\beta$
Protein G	1PGA	56	$1\alpha + 4\beta$
Protein G Variant	NUG1	57	$1\alpha + 3\beta$
Protein G Variant	NUG2	57	$1\alpha + 3\beta$
Alpha-Spectrin SH3 Domain	1SHG	57	$1\alpha + 5\beta$
Human FYN	1NYF	58	$5\alpha + 1\beta$
Immunoglobulin G Binding Protein A	2SPZ	58	3α
Cardiotoxin III	2CRS	60	5β
Tick Antocoagulant peptide	1TCP	60	$2\alpha + 2\beta$
ADR1	2ADR	60	$2\alpha + 2\beta$
Repressor Protein C1	1R69	63	5α
Chymotrypsin Inhibitor 2 variant	1COA	64	$1\alpha + 4\beta$
Chymotrypsin Inhibitor 2 variant	2CI2	65	$1\alpha + 4\beta$
Probable enterotoxin	2KRS	70	7β
Regulatory Protein CRO	2CRO	71	5α
Protein L	2PTL	78	$1\alpha + 4\beta$
Procarboxy peptidase B	1PBA	81	$4\alpha + 3\beta$
Procarboxy peptidase A2	106X	81	$2\alpha + 3\beta$
ACYL-CO Enzyme	2ABD	86	4α
Barnase	1YVS	106	$3\alpha + 4\beta$
Binase	1BUJ	109	$5\alpha + 3\beta$
DNA B Helicase	1JWE	114	8α

data. We examine shortest paths from all unfolded states to the native state. (Recall that roadmap edge weights reflect the transition's energetic feasibility, so extracting the smallest weighted path corresponds to extracting the most energetically feasible path). We then compare the dominant ordering (i.e., the ordering that occurs most frequently among all folding pathways present) to the ordering given by experimental data.

- *Pathway Quality:* We define folding pathway quality as the weight of each edge (i.e., its energetic feasibility) multiplied by the dominance of that edge (i.e.,

the number of folding pathways that traverse it). This metric is important because it identifies how many edges with low energies are present and how frequently they are used. Having low quality values in our results indicates a better performing connection method.

Experiment Setup for Sampling

For all learning during sampling experiments, we generate conformations using the different variants of the sampling method based on rigidity analysis [103] as described in Section 5.1.3.1. We make changes to the P_{flex} (perturb flexible torsional angles), and P_{rigid} (perturb rigid torsional angles) parameters. The range for P_{flex} and P_{rigid} increases in flexibility and rigidity as we go from 0 to 1.

Table 5.2 gives a list of the variants to the sampling methods we utilize.

Table 5.2: Rigidity Sampling Variants used

Sampling Variant	P_{flex}	P_{rigid}
Rigidity-1	0.2	0.2
Rigidity-2	0.8	0.2
Rigidity-3	0.2	0.8
Rigidity-4	0.1	1.0
Rigidity-5	1.0	1.0

We use a combination of our KClosest neighbor finder using the cluster distance metric and the straight line local planner as our connection method for all cases and attempt to connect to 20 nearest neighbors. For LLC, we set NF_{local} to be the 40 nearest neighbors based on Euclidean distance. We stop construction once we have a stable roadmap.

Experiment Setup for Connection

For all learning during connection experiments, we generate conformations using iterative sampling based on rigidity analysis [103]. For all connection methods, we use a straight line local planner and attempt to connect to the 20 nearest neighbors. For local learning, we set NF_{local} to be the 40 nearest neighbors based on Euclidean distance. This resulted in the best performance in preliminary experiments. We stop construction once we have a stable roadmap.

5.3.1 Learning in Sampling

In this section, we investigate the performance of LLS (local learning), GLS (global learning), and individual sampling methods to model the folding landscape of 23 proteins. GLS and LLS use these methods as their learning set.

We first establish each method's ability (individual sampling methods, global learning, and local learning) to validate against experimental data when available. We examine the quality of the resulting folding pathways and the time required by each individual method and look at the cumulative performance of these metrics. We show how LLS's learning decisions are consistent with the individual connection method performance outside of the learning framework. In addition, we compare LLS's learning performance against GLS's learning approach.

5.3.1.1 Validation by Secondary Structure Formation Order

Table 5.3 summarizes the comparison of each method's dominant secondary structure formation order. (Entries are ordered as appears in Table 5.1 by protein length.) Only the learning methods (LLS and GLS) produced the same dominant formation order as experiment for all proteins with available data. Individual methods were

unable to reproduce the ordering from experimental data for 2CI2. Thus, in some cases learning was required for correctness.

Table 5.3: Learning in Sampling:Validation of secondary structure formation order to experimental data when available. Proteins are ordered by protein length as in Table 5.1.

PDB Identifier	Experimental Data	LLS	GLS	Rigidity-1	Rigidity-2	Rigidity-3	Rigidity-4	Rigidity-5
1RDV	unavailable			same ordering				
1FCA	unavailable			same ordering				
1PGA	[62]	Y	Y	Y	Y	Y	Y	Y
NUG1	[89]	Y	Y	Y	Y	Y	Y	Y
NUG2	[89]	Y	Y	Y	Y	Y	Y	Y
1SHG	[74, 105]	Y	Y	Y	Y	Y	Y	Y
1NYF	[44, 92]	Y	Y	Y	Y	Y	Y	Y
2SPZ	unavailable			same ordering				
2CRS	[70]	Y	Y	Y	Y	Y	Y	Y
1TCP	unavailable			same ordering				
2ADR	unavailable			**different orderings**				
1R69	unavailable			same ordering				
1COA	unavailable			same ordering				
2CI2	[49]	Y	Y	N	N	N	N	N
2KRS	[74]	Y	Y	Y	Y	Y	Y	Y
2CRO	unavailable			same ordering				
2PTL	[111]	Y	Y	Y	Y	Y	Y	Y
1PBA	unavailable			same ordering				
106X	[106]	Y	Y	Y	Y	Y	Y	Y
2ABD	[101]	Y	Y	Y	Y	Y	Y	Y
1YVS	[75]	Y	Y	Y	Y	Y	Y	Y
1BUJ	unavailable			**different orderings**				
1JWE	unavailable			same ordering				
# Agree with Exp. / # Available		12/12	12/12	11/12	11/12	11/12	11/12	11/12

When experimental data was not available, all methods produced the same ordering for 9 proteins and different orderings for 2 proteins (2ADR and 1BUJ). For 2ADR, GLS and LLS produce similar orderings with Rigidity-1, Rigidity-2 and Rigidity-5

and is different from the ordering produced with Rigidity-3 and Rigidity-4. The difference in the ordering is noticed in the last α helix and β sheet ordering where there is a switch in the ordering when compared. For 1BUJ, all the methods produced similar orderings apart from Rigidity-4 which disagree in the ordering of the first α helix.

5.3.2 Quality, Time, and the Tradeoff Between Them

Quality

Figure 5.1 shows the folding pathway quality using the different variants of the rigidity sampling method, GLS and LLS and we see that LLS outperforms all the other methods including GLS for 21 out of the 23 proteins studied. Rigidity-1 and Rigidity-5 is best for one protein only i.e., 1NYF and 1JWE respectively. Apart from the best choice being LLS, other methods don't consistently produce good quality for all the proteins studied. When we look at the individual sampling method performance, we see that no single method performs best across all proteins which shows a clear need for learning. Also when we consider the correlation we observe that as we increase in the size of the protein, the pathway quality does not diminish using LLS as compared to Rigidity-3 which indicates an improvement when learning is employed.

Time

Figure 5.2 provides the time needed to build a stable roadmap using the different sampling methods. LLS is the best for four of the proteins and second best for five, with two of them incurring less than 5% overhead. LLS improves over GLS for 21 out ofthe 23 proteins and in some cases with a 50% improvement. We see here the same trend where no particular method is the best for all the proteins. Rigidity-1

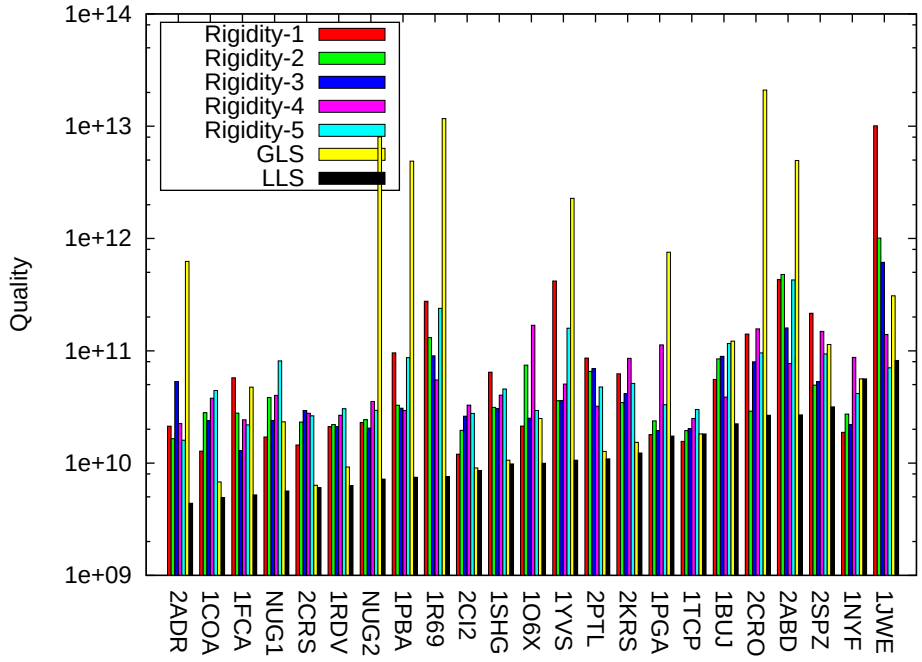


Figure 5.1: Sampling Results over Quality.

performs best for 11 of the proteins but not so well when we consider the quality of pathways produced as seen in Figure 5.1. LLS gives a good balance between the time and the quality.

Exploring in more detail on the performance of LLS in relation to the other methods, we plot the difference in time between LLS and Rigidity-1 (better non-learned sampling method) as a function of protein length. The larger difference presented in Figure 5.3 shows a better quality recorded for LLS because the lower the energy recorded in our pathways the better. Our results show clearly that irrespective of the length we obtain better quality for 22 out of the 23 proteins studied. Figure 5.4 shows the difference in time between Rigidity-1 and LLS, the higher the difference the better for LLS in terms of performance. Results indicate that we have a varying performance using LLS, but when we compare Figure 5.3 and Figure 5.4 we see that

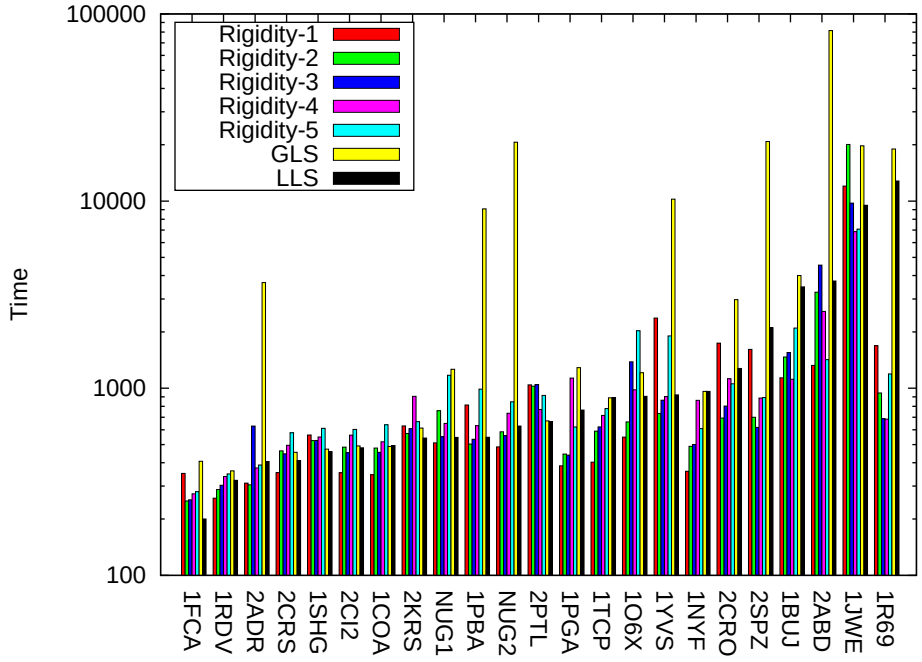


Figure 5.2: Sampling Results over Time.

LLS balances between the time and quality.

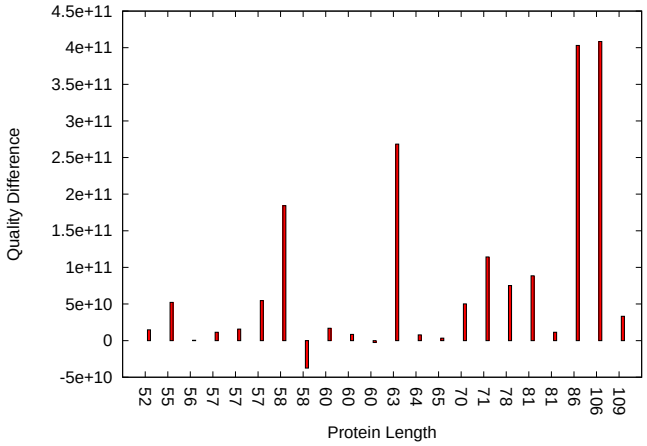


Figure 5.3: Sampling Results over Linear Correlation with Quality

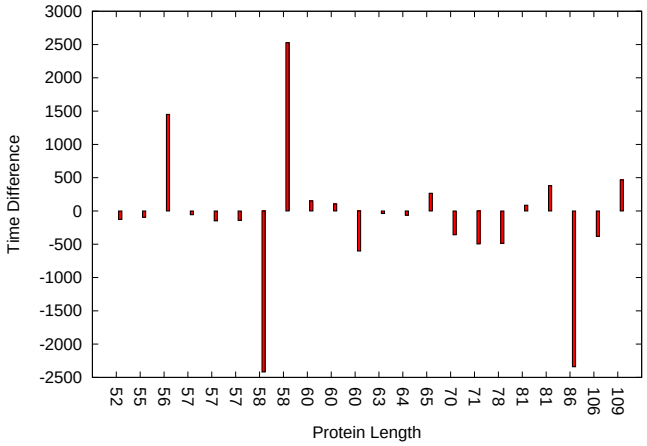


Figure 5.4: Sampling Results showing Linear Correlation over Time

5.3.2.1 Quality vs. Time

Another important analysis we perform is to study the cumulative performance across all 23 proteins studied. Figure 5.5 shows the ordered ranking of each connection method, GLS, and LLS across all 23 proteins. For each protein, we assign a rank from one to seven (with seven being the best) to each method for quality and time. The cumulative performance for each method is the average of these rankings. Here we see that LLS has the best cumulative quality performance of 6.8 across the entire environments even with Rigidity-2 performing best with regards to time.

Figure 5.5 gives more insight into the behavior of the different methods. We plot the quality of LLS and Rigidity-1 which is the best performing method in terms of time. Here we see that there is little change in performance of LLS as the protein length increases which cannot be said for Rigidity-1 which shows a loss in quality as the protein length increases. This shows that LLS is more scalable in providing quality pathways which in turn affects the ability to simulate the folding process as observed by biochemists.

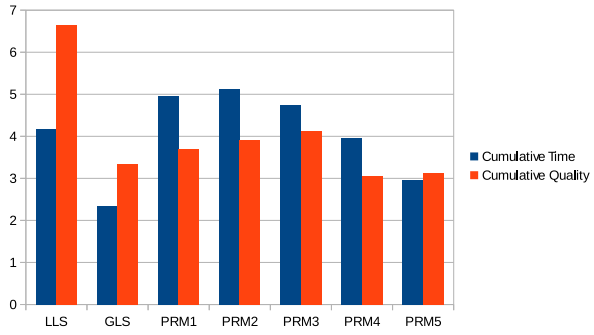


Figure 5.5: Cumulative Results for Sampling

5.3.2.2 Inspection of LLS Learning Choices

Figure 5.6 shows the percentage at which LLS used each individual connection method in constructing stable roadmaps for each protein. LLS's use of the different methods for protein 1FCA and ITCP is dominated by Rigidity-1, NUG1 and IJWE by Rigidity-2, 1O6X and 1YVS by Rigidity-4 and 2CI2, 2KRS, 2PTL, IPGA, and 1NYF by Rigidity-5 while for the other proteins we have a mixture of the different sampling methods being used. When it favors a set of methods, it favors the best individual method in terms of time and quality twice and favors the best method in terms of time nine times of the total proteins studied.

5.3.3 Learning in Connection

In this section, we investigate the performance of LLC (local learning), GLC (global learning), and individual connection methods to model the folding landscape of 23 proteins. Individual connection methods are K-Closest neighbor selection using either Cluster, Euclidean, or lRMSD distance metric. GLC and LLC use these methods as their learning set.

We first establish each method's ability (individual connection methods, global learn-

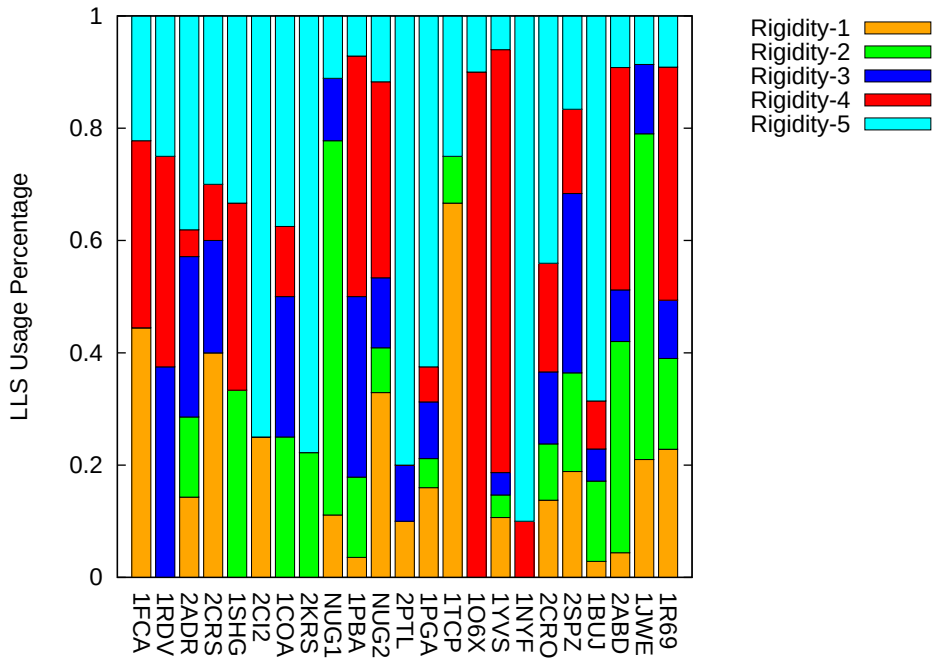


Figure 5.6: Sampling Results Local Use

ing, and local learning) to validate against experimental data when available. We then look into the local planner success rate in the context of each strategy. We examine the quality of the resulting folding pathways and the time required by each individual method and look at the cumulative performance of these metrics. We show how LLC's learning decisions are consistent with the individual connection method performance outside of the learning framework. In addition, we compare LLC's learning performance against GLC's learning approach.

5.3.3.1 Validation by Secondary Structure Formation Order

Table 5.4 summarizes the comparison of each method's dominant secondary structure formation order. (Entries are ordered as appears in Table 5.1 by protein length.) Only the learning methods (GLC and LLC) produced the same dominant formation

order as experiment for all proteins with available data. Individual methods were unable to reproduce the ordering from experimental data for 2ABD. Thus, in some cases learning was required for correctness.

Table 5.4: Learning in Connection:Validation of secondary structure formation order to experimental data when available. Proteins are ordered by protein length as in Table 5.1.

PDB Identifier	Experimental Data	LLC	GLC	Cluster	Euclidean	lRMSD
1RDV	unavailable			same ordering		
1FCA	unavailable			same ordering		
1PGA	[62]	Y	Y	Y	Y	Y
NUG1	[89]	Y	Y	Y	Y	Y
NUG2	[89]	Y	Y	Y	Y	Y
1SHG	[74, 105]	Y	Y	Y	Y	Y
1NYF	[44, 92]	Y	Y	Y	Y	Y
2SPZ	unavailable			**different orderings**		
2CRS	[70]	Y	Y	Y	Y	Y
1TCP	unavailable			same ordering		
2ADR	unavailable			same ordering		
1R69	unavailable			same ordering		
1COA	unavailable			same ordering		
2CI2	[49]	Y	Y	Y	Y	Y
2KRS	[74]	Y	Y	Y	Y	Y
2CRO	unavailable			same ordering		
2PTL	[111]	Y	Y	Y	Y	Y
1PBA	unavailable			same ordering		
106X	[106]	Y	Y	Y	Y	Y
2ABD	[101]	Y	Y	**N**	**N**	**N**
1YVS	[75]	Y	Y	Y	Y	Y
1BUJ	unavailable			**different orderings**		
1JWE	unavailable			same ordering		
# Agree with Exp. / # Available		12/12	12/12	11/12	11/12	11/12

When experimental data was not available, all methods produced the same ordering for 9 proteins and different orderings for two proteins (2SPZ and 1BUJ). Upon examination of the two proteins that methods disagree on, we find that LLC, GLC, and Cluster are always in agreement and Euclidean and lRMSD are always in agreement. Additionally, disagreements only occur at the end of the pathway; all methods agree on the order of the first elements to form. Specifically, all methods find that the central α-helix forms first in 2SPZ and disagree on the relative ordering of the two terminal α-helices. Similarly, all methods find that β-strands 6, 5, 4, 3, 2 form first (and in that order) and disagree on the relative ordering of the three α-helices and the remaining β-strand for 1BUJ.

5.3.3.2 Local Planner Success Rate

Recall that a connection method comprises both the distance metric used to identify neighbors to connect and a local planner (e.g., a straight-line in $\phi - \psi$ space) that computes a set of intermediate conformations, evaluates their energetic viability, and adds an edge between the two neighbors if such a trajectory is feasible. The local planner success rate is a good indicator of the performance of the entire connection process. We measure the local planner success rate as the number of connections made out of the number of connections attempted.

Figure 5.7 displays the local planner success rate for all connection methods across all proteins studied. Observe that the local planner success rate is highest for LLC for 18 of the 23 proteins and comparable for one of the proteins (1RDV). For proteins in which it is not the highest (1NYF, 1PGA, 2ADR, 2CRS), it is within 0.05 of the highest. Note that GLC does not perform as well as LLC and in many cases (for 15 proteins it is greater than 0.1 lower) is significantly lower. This indicates that not only

is learning important, but *local* learning is crucial to properly adapting to different protein folding landscapes. LLC consistently makes wise choices for connection that yield successful local planner attempts, which are quite costly.

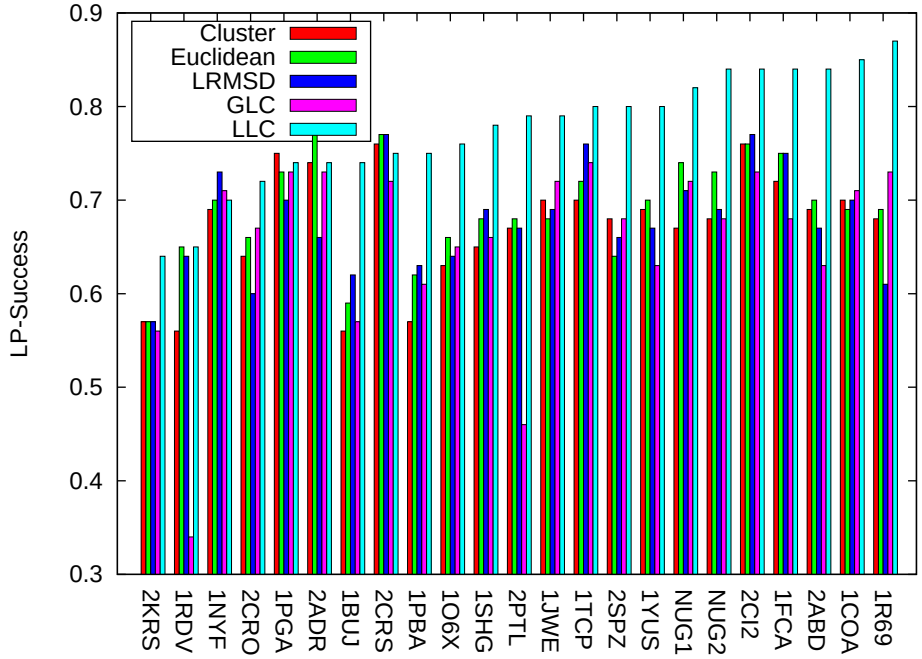


Figure 5.7: Local planner success rate for each method over all proteins studied. The local planner success rate of LLC is greater than all the other methods for 18 of the 23 proteins studied and comparable for 1 of the proteins. Note that entries are ordered by the local planner success rate in the context of LLC.

5.3.3.3 Quality, Time, and the Tradeoff Between Them

Quality

Figure 5.8 shows the folding pathway quality of each connection method, GLC, and LLC. Entries are ordered by LLC performance (and not by protein length). Recall that the aim is to generate pathways with low weight/energy. Only looking

at individual connection method performance, we first see that no single connection method performs the best across all proteins: Cluster is the best choice for seven proteins, Euclidean for 11 proteins, and lRMSD for five proteins. In addition, there is no correlation between individual connection method performance and secondary structure makeup or size. Thus, there is a clear need for learning.

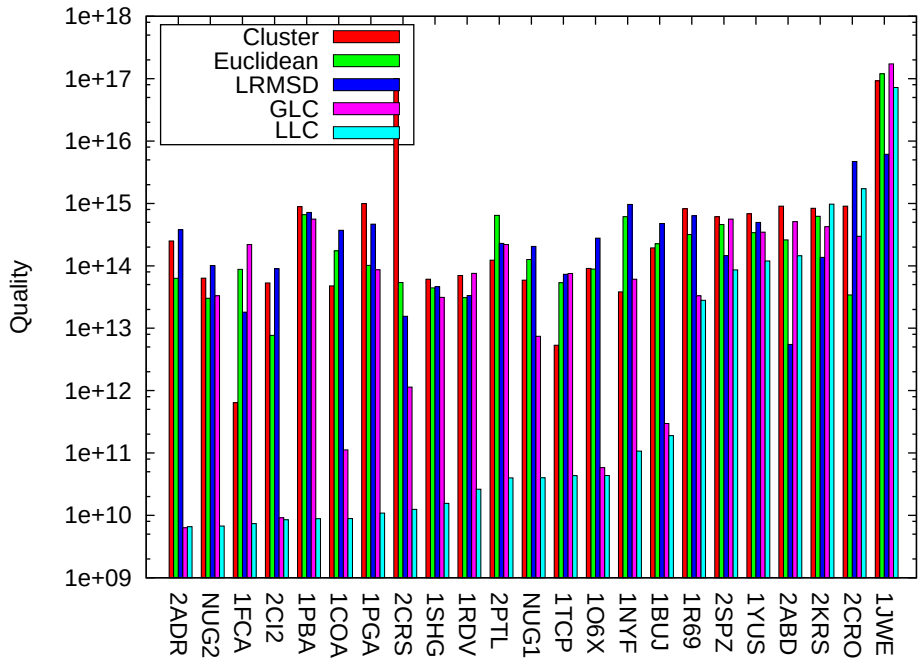


Figure 5.8: Roadmap quality for each method over all proteins studied. No single individual connection method performs best across all proteins. LLC produces the best quality roadmaps for 18 of the 12 proteins studied. Note that entries are ordered by LLC performance.

It is not surprising then that learning methods outperform the best individual connection methods much of the time: GLC (pink bars) produces lower weighted pathways than Cluster, Euclidean, and lRMSD for 11 of the 23 proteins, and LLC (blue bars) for 19 of the 23 proteins. Notice, however, that the type of learning is important.

LLC with its local learning is much more successful than GLC with its global approach. GLC outperforms LLC for only one protein in the set (2ADR) and even then the performance is only marginally better while LLC outperforms GLC by a large margin for many of the proteins. In fact, LLC is the best approach for 18 out of the 23 proteins studied. Note that the best performing method in the other five proteins is not the same (many of them are at the far right of Figure 5.8): lRMSD produces lower weight pathways for three proteins (2KRS, 2ABD, and 1JWE), Euclidean for one (2CRO), and GLC for one (2ADR).

Additionally, in 17 of the 18 proteins where LLC produces the best quality, it produces significantly better quality than the other methods for 12 of the 18. We see an improvement of LLC over GLC in terms of quality for 20 of the 23 proteins studied. Of the three remaining proteins (2ADR, 2CRO, 2KRS) where GLC perform.

Time

Figure 5.9 provides the time needed to build stable roadmaps for each method, ordered by protein length. LLC is the fastest for six of the proteins and the second fastest for six, with three of those incurring less than 10% overhead. Thus, LLC performs as well as or better than the best performing method for 12 out of 23 proteins (52% of the time), while GLC performs best for only three. Just as with quality, the best performing individual connection method varies between proteins: Euclidean is fastest for 11 proteins, Cluster for two, and lRMSD for one. Euclidean is most often the fastest method but is the best method in terms of quality for only one protein.

To further understand the scalability of these approaches, we plot the time to build a stable roadmap as a function of protein length for both LLC and its fastest competitor, Euclidean. Each point in Figure 5.10 corresponds to the time taken for a

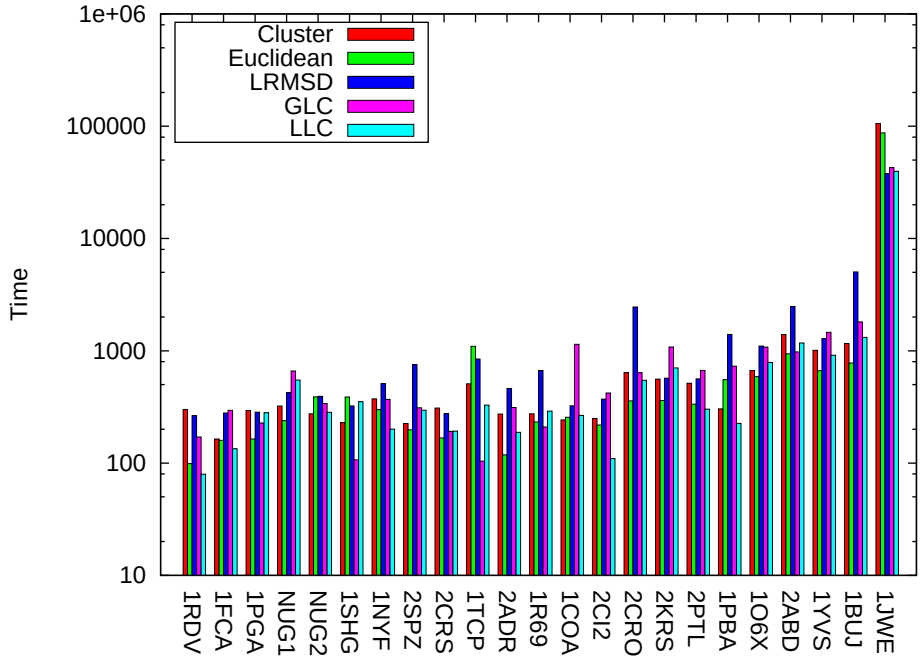


Figure 5.9: Time for each method over all proteins studied. LLC performs as well as or better than the best performing method for 12 out of 23 proteins studied. Note that entries are ordered by protein length.

protein of that length. Figure 5.10 also plots a linear regression for each data set. There is a roughly linear relationship between length and running time (correlation coefficients of 0.55 for LLC and 0.53 for Euclidean; higher polynomial regressions fit poorly). Note that while we see some overhead for learning (i.e., a steeper regression line), other methods may not produce pathways of high quality. For example, ACYL-CO Enzyme (2ABD) is a protein where only LLC produced the correct secondary structure formation order as seen in experiment (see Table 5.4). It is also the furthest outlier above the regression line (length 86). While more time is consumed constructing a stable roadmap for this protein, it is time well-spent as it produces the correct secondary structure formation order while others do not.

We also consider the difference in time between LLC and Euclidean as shown in

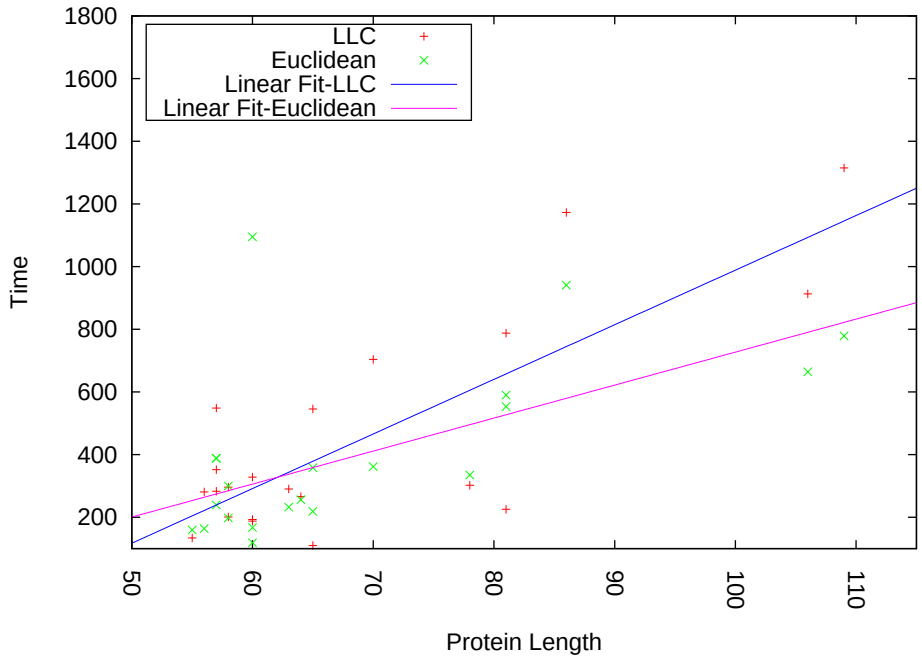


Figure 5.10: Time as a function of protein length. LLC and its fastest competitor, Euclidean, display a roughly linear relationship between time and protein length.

Figure 5.11. Here, just as we saw for learning during sampling, we have varying performance across the proteins irrespective of length but with more importance on quality, we still produce roadmaps within an acceptable time period.

5.3.3.4 Quality vs. Time

Finally, we look at each method's cumulative performance to examine how these two metrics interplay. Figure 5.12 shows the ordered ranking of each connection method, GLC, and LLC across all 23 proteins. For each protein, we assign a rank from one to five (with five being the best) to each method for quality and time. The cumulative performance for each method is the average of these rankings.

LLC performs better than the other connection methods across the entire protein set in terms of quality and second best in terms of time. lRMSD, as expected, is

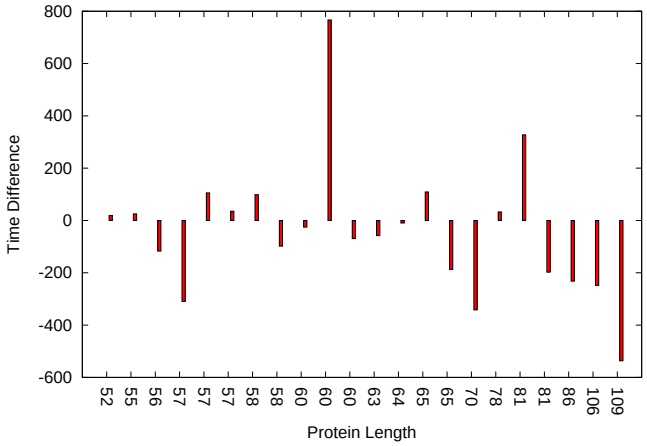


Figure 5.11: Difference in time between LLC and its fastest competitor Euclidean.

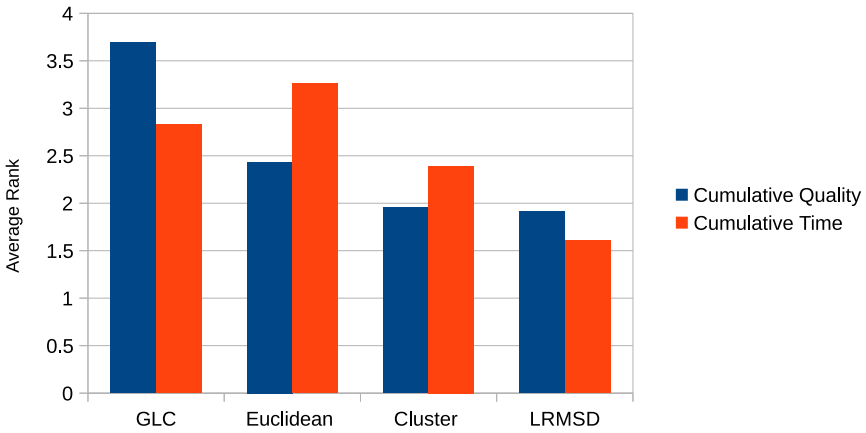


Figure 5.12: Cumulative performance of each method over all proteins studied. Methods are ranked from one (worst) to five (best). Entries are ordered by cumulative quality ranking. LLC performs better than the other methods across the entire protein set in terms of quality and second best in terms of time.

the slowest. While LLC is not the fastest overall (Euclidean is), it does produce the best quality. LLC is the only method that is able to adapt locally to varying energy landscapes and thus yields higher quality roadmaps. GLC is the second best in terms

of quality but third in terms of time. LLC outperforms GLC.

Figure 5.13 compares the difference in quality of LLC to the quality of the fastest competitor, Euclidean. We see that regardless of protein length, LLC consistently outperforms Euclidean in terms of quality for most of the proteins studied. Recall that the aim is to generate pathways with low weight/energy. While computation time is important (and we have shown that LLC is competitive with other methods in this regard), it is more important to produce pathways of higher quality.

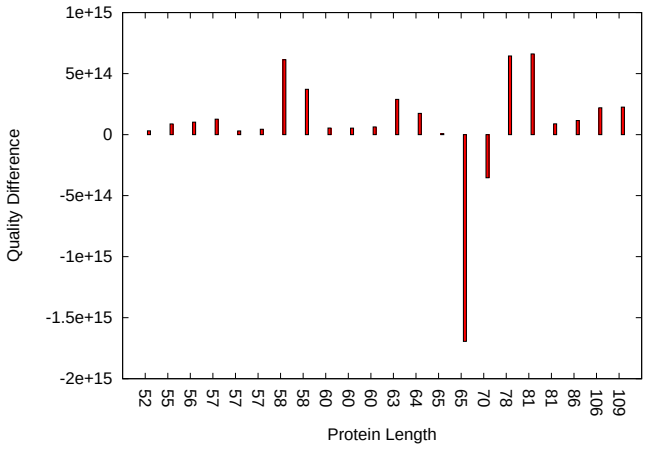


Figure 5.13: Quality as a function of protein length. LLC outperforms and its fastest competitor, Euclidean, in terms of quality irrespective of protein length.

5.3.3.5 Inspection of LLC Learning Choices

Figure 5.14 shows the percentage at which LLC used each individual connection method in constructing stable roadmaps for each protein. Entries are ordered by Euclidean usage as it is most often selected across the entire set.

For many proteins, LLC favors a single connection method, but for some (1O6X,

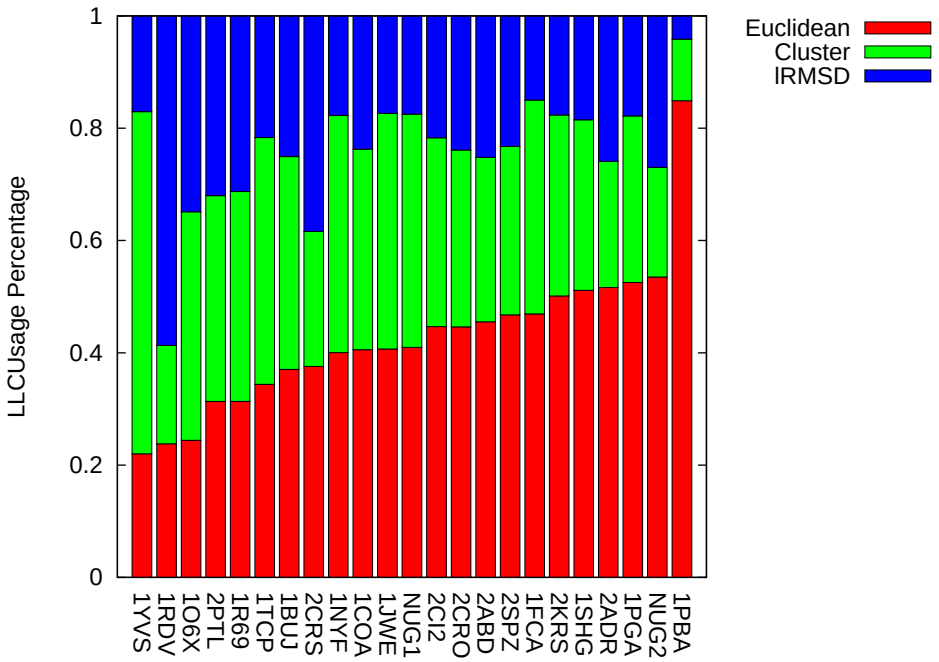


Figure 5.14: Connection method usage percentage in LLC across all proteins studied. Entries ordered by Euclidean usage.

1TCP and NUG1), it favors two connection methods, and for two proteins (2PTL and 1R69), it selects equally among all connection methods. When it favors a subset of the connection methods, it selects the best individual method in both time and quality for nine proteins, the best individual method in time only for four proteins, and the best individual method in quality only for three proteins.

5.4 Heterogeneity of the C-Space for Proteins

Considering the environment for motion planning for a robot, one can make a relatively straight forward analogy of the C-Space based on the workspace representation of the environment. We can determine how heterogeneous an environment is by considering the visibility representations in the C-Space to determine narrow passages and obstacles in the environment. For proteins however, this becomes difficult be-

cause it is difficult to map the energy landscape of proteins and view visually.

When we consider Figure 5.1 and 5.2 and the different performance of the different sampling methods for a given protein we see intuitively the heterogeneity characteristics of the C-Space for protein folding. In this section we explore this heterogeneity by examining distances between parent and child configurations and potential energies of the protein configurations forming nodes in the roadmap.

5.4.1 Distance between Parent and Child Configurations

In heterogeneous environments, the ability to generate valid nodes will vary in different regions of the landscape. Thus, one metric that could be used to measure heterogeneity might be the variances in the distances between parents (original configurations) and their children (the new configurations generated by perturbing the parent configuration). In particular, in a purely homogeneous environment the distance between a parent and child would be uniform throughout the landscape. In a heterogeneous environment, however, the distance between parents and children may vary, and this could be observed in multiple ways. For example, given configurations a and b, if the average distance from a to its children is different from the average distance from b to its children, then this is an indication that the environment is heterogeneous and moreover, that a and b are in different regions of the environment. As another example, if a configuration a has several child configurations, and if there is a large variance in the distances to those children, then this is also an indication of heterogeneity in the folding landscape.

Figure 5.15 gives such a scenario, we use Protein G (1GB1) as an example case study. The plot shows the average distance between a parent configuration and it's children in different regions. We see varying distances, indicating that our protein

environment is indeed heterogeneous.

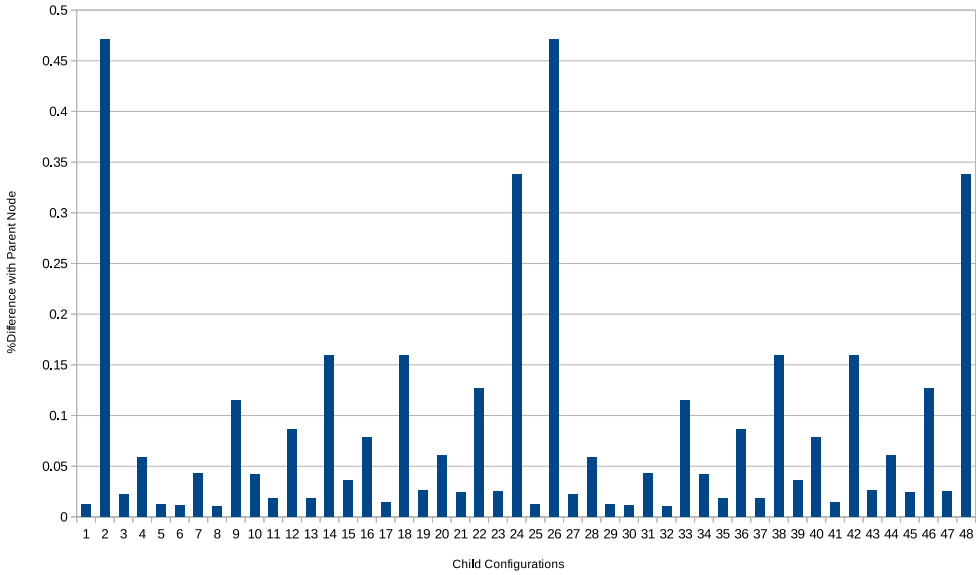


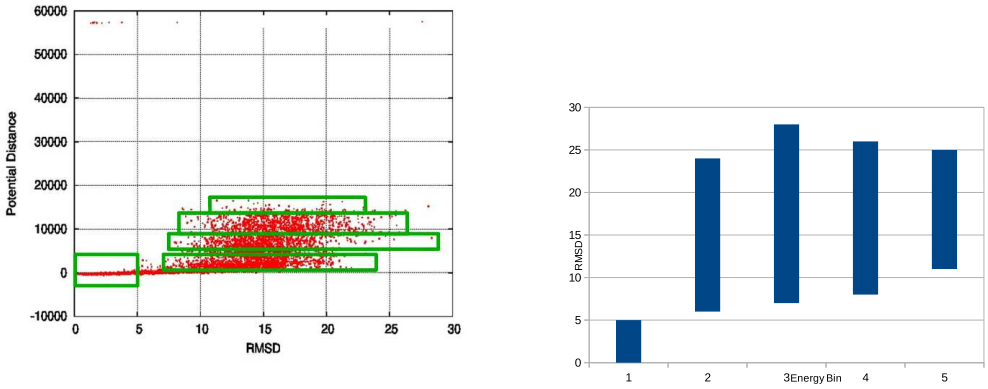
Figure 5.15: Heterogeneity: Difference between the parent and children configurations

5.4.2 Potential vs RMSD and Heterogeneity

To further show the heterogeneous of the C-Space for proteins, we make plots for three of our studied proteins, an α protein 2ABD, a β protein 2CRS and an α, β mix protein 1GB1. We show the relationship for configurations based on the potential energy and the RMSD of the intermediate configurations for these proteins while building the roadmap.

Proteins containing only α proteins form their secondary structure helices one at a time and then assemble them to form the tertiary structure. This is reflected in Figure 5.16(a) where the narrow tail produced indicates small changes in energy as RMSD approaches zero [6]. Proteins containing only β secondary structures form

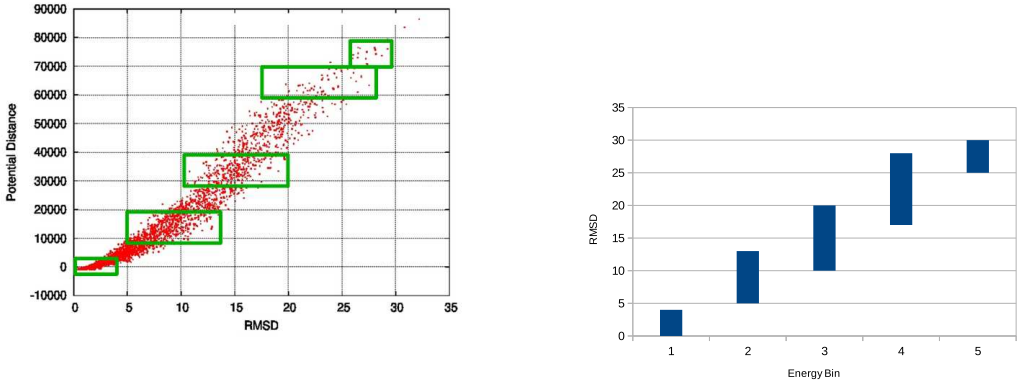
both their secondary and tertiary structure at the same time which is indicated in the smoother plots in Figure 5.17(a) while proteins with a mix of α and β proteins give a variation of both characteristics as shown in Figure 5.18(a).



(a) Potential-RMSD (b) RMSD-Energy Bin

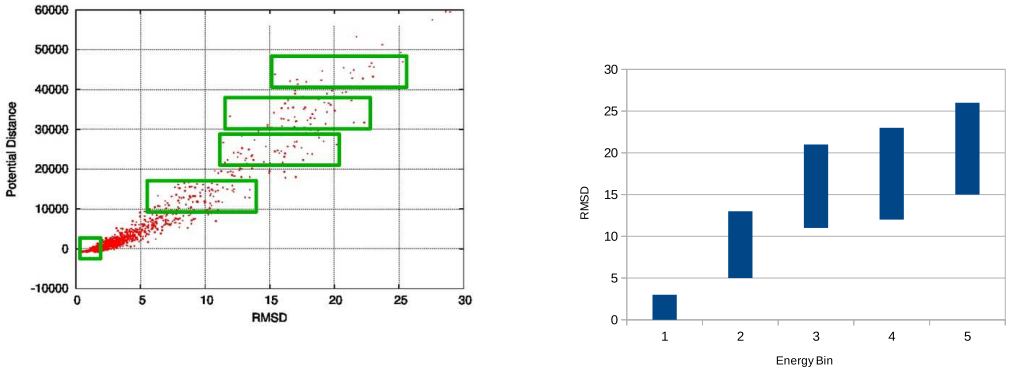
Figure 5.16: Heterogeneity: An α only protein (2ABD)

Using this information to investigate the heterogeneity of the C-Space for proteins, we create bin representations shown in green in all our Potential-RMSD plots. The bins represent intermediate configurations grouped based on RMSD distance and potential energy and give an idea of our configuration space or energy landscape in the case for proteins. We see by looking at Figure 5.16(b), 5.17(b) and 5.18(b) that there is a variance in the size of these intermediate configurations present which shows a clear indication about the heterogeneity of our C-Space for proteins.



(a) Potential-RMSD (b) RMSD-Energy Bin

Figure 5.17: Heterogeneity: A β only protein (2CRS)



(a) Potential-RMSD (b) RMSD-Energy Bin

Figure 5.18: Heterogeneity: A Mix(α, β) protein (1PGA)

6. CONCLUSION

In this work, we present an approach that uses local learning to select appropriate sampling and connection methods methods in the context of PRM roadmap construction for robot motion planning and protein folding. Our method monitors the performance and cost of various methods within the local neighborhood of the sampling or connecting conformation and adjusts their selection probabilities accordingly based on a cumulative reward approach. We performed experiments in different heterogeneous environments ranging from 2D to 3D environments and real world scenarios. We showed a need for learning based on the improvement in performance when it was applied. In particular we show the advantage of being able to dynamically partition the environment intelligently on the fly.

Table 6.1: Lessons Learned

	Local Sampling	Global Sampling
Local Connection	Highly Heterogeneous High DOF problems High geometric complexity Longer time to solve query	Moderately Homogeneous Medium to Low DOF problems
Global Connection	Moderately Homogeneous Medium to High DOF problems	Highly Homogeneous Low DOF problems Simple 2D environments Shorter time to solve query

Table 6.1 gives a summary of lessons learned during our robot motion planning experiments. We give some insight into when different learning combinations are most beneficial, considering local and global learning for sampling and connection. Based

on our experiments, local learning during both sampling and connection benefits high DOF problems, high geometric complexity problems and environments that are difficult to plan within a time limitation.

In protein folding simulations, we have demonstrated a clear need for learning (i.e., our method was the only method to validate against all available experimental data) and showed that local learning is superior to global learning (i.e., local learning outperformed all other methods in terms of quality for 18 out of 23 proteins and was either the fastest or second fastest for 12 of the proteins). We also showed that our method produced a higher local planner success rate. In many cases, local learning produced significantly higher quality results than the other methods.

In summary, we proposed a novel method of learning localized to regions and show its superiority to traditional and current state of the art sampling-based motion planning techniques. Local learning removes the need to explicitly partition the environment and the burden of deciding which method to use. It leverages the strengths of the individual input methods, and is extendable to include other future planning methods. Finally, based on our experiments and evaluation both in robotics application and protein folding simulations, we provide guidance on how to apply our framework to any motion planning problem.

REFERENCES

[1] Pieter Abbeel, Dimitri Dolgov, Andrew Y. Ng, and Sebastian Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1083–1090, Nice, France, September 2008. IEEE.

[2] J. M. Ahuactzin and K. Gupta. A motion planning based approach for inverse kinematics of redundant robots: The kinematic roadmap. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 3609–3614, 1997.

[3] Ibrahim Al-Bluwi, Thierry Simeon, and Juan Cortes. Motion planning for molecular simulations; a survey. *Computer Science Review*, 6(4):125–143, 2012.

[4] Ibrahim Al-Bluwi, Marc Vaisset, Thierry Siméon, and Juan Cortés. Modeling protein conformational transitions by a combination of coarse-grained normal mode analysis and robotics-inspired methods. *BMC Structural Biology*, 13(1):1–14, 2013.

[5] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

[6] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2001.

[7] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 113–120, 1996.

[8] Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo. OBPRM: an obstacle-based PRM for 3d workspaces. In *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd. (WAFR '98).

[9] M. S. Apaydin, D.L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. *J. Comput. Biol.*, 10(3–4):257–281, 2004. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2004.

[10] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[11] Peter Auer, Nicolò Cesa-bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2003.

[12] Adam L. Beberg, Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq, and Vijay S. Pande. Folding@Home: Lessons from eight years of volunteer distributed computing. In *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8, Atlanta, Georgia, April 2009. IEEE Computer Society.

[13] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3671–3678, Saint Paul, Minnesota, 2012. IEEE.

[14] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[15] Joshua Bialkowski, Sertac Karaman, and Emilio Frazzoli. Massively parallelizing the RRT and the RRT*. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2011.

[16] R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[17] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, May 1999.

[18] R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. Int. Conf. Artif. Intel.*, pages 799–806, 1983.

[19] Joseph D. Bryngelson and Peter G. Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proc. Natl. Acad. Sci. USA*, 84:7524–7528, 1987.

[20] Sabastien Bubeck, Rami Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *In proceedings of the twentieth International Conference on Algorithmic Learning Thoemry (ALT 2009)*, pages 23–37, 2009.

[21] Brendan Burns and Oliver Brock. Sampling-based motion planning using predictive models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3313–3318, 2005.

[22] Brendan Burns and Oliver Brock. Toward optimal configuration space sampling. In *Proc. Robotics: Sci. Sys. (RSS)*, pages 105–112, 2005.

[23] F. Chiti and C. M. Dobson. Protein misfolding, functional amyloid, and human disease. *Annu. Rev. Biochem.*, 75:333–366, 2006.

[24] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[25] Hsin-Yi Yeh (Cindy), Shawna L. Thomas, David Eppstein, and Nancy M. Amato. UOBPRM: A uniformly distributed obstacle-based PRM. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 2655–2662, 2012.

[26] J. Cortés, T. Siméon, M. Remaud-Siméon, and V. Tran. Geometric algorithms for the conformational analysis of long protein loops. *J. Computat. Chem.*, 25(7):956–967, 2004.

[27] David G. Covell. Folding protein α-carbon chains into compact forms by Monte Carlo methods. *Proteins: Struct. Funct. Bioinf.*, 14(3):409–420, 1992.

[28] L. K. Dale and N. M. Amato. Probabilistic roadmaps – putting it all together. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1940–1947, 2001.

[29] Jory Denny and Nancy M. Amato. Toggle PRM: simultaneous mapping of c-free and c-obstacle - a study in 2d -. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pages 2632–2639, 2011.

[30] Jory Denny and Nancy M. Amato. The toggle local planner for sampling-based motion planning. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 1779–1786, 2012.

[31] Jory Denny and Nancy M. Amato. Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension. In *Algorithmic Foundations of Robotics X*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 297–312. Springer, Berlin/Heidelberg, 2013. (WAFR '12).

[32] Jory Denny, Evan Greco, Shawna Thomas, and Nancy M. Amato. MARRT: Medial axis biased rapidly-exploring random trees. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 90–97, Hong Kong, China, May 2014.

[33] Jory Denny, Marco Morales, Samuel Rodriguez, and Nancy M. Amato. Adapting RRT growth for heterogeneous environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1772–1778, Tokyo, Japan, November 2013.

[34] W. A. Eaton, V. Muñoz, P. A. Thompson, C. Chan, and J. Hofrichter. Submillisecond kinetics of protein folding. *Curr. Op. Str. Biol.*, 7:10–14, 1997.

[35] Chinwe Ekenna, Sam Ade Jacobs, Shawna Thomas, and Nancy M. Amato. Adaptive neighbor connection for PRMs: A natural fit for heterogeneous environments and parallelism. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1249–1256, Tokyo, Japan, November 2013. IEEE.

[36] Chinwe Ekenna, Shawna Thomas, and Nancy M. Amato. Adaptive local learning in sampling based motion planning for protein folding. In *The IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 61–68, October 2015.

[37] Chinwe Ekenna, Diane Uwacu, Shawna Thomas, and Nancy M. Amato. Improved roadmap connection via local learning for sampling based planners. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 3227–3234, Hamburg, Germany, October 2015.

[38] Chinwe Ekenna, Diane Uwacu, Shawna Thomas, and Nancy M. Amato. Improved roadmap connection via local learning for sampling based planners. Technical Report TR15-006, Texas A&M, April 2015.

[39] Chinwe Ekenna, Diane Uwacu, Shawna Thomas, and Nancy M. Amato. Studying learning techniques in different phases of prm construction. In *Machine Learning in Planning and Control of Robot Motion Workshop (IROS-MLPC)*, Hamburg, Germany, October 2015.

[40] Mikhail Frank, Jurgen Leitner, Marijn Stollenga, Alexander Forster, and Jurgen Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in Neurorobotics*, 7(25), 2014.

[41] A. Gareth, Michele Ceriotti, and Michele Parrinello. A self-learning algorithm for biased molecular dynamics. *Proc. Natl. Acad. Sci. USA*, 107(41):17509–17514, 2010.

[42] Russell Gayle, Ming C. Lin, and Dinesh Manocha. Constraint-based motion planning of deformable robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1046–1053, April 2005.

[43] R. Geraerts and M. H. Overmars. Reachablility analysis of sampling based planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 406–412, 2005.

[44] Viara P. Grantcharova, David S. Riddle, Jed V. Santiago, and David Baker. Important role of hydrogen bonds in structurally polarized transition state folding of the src SH3 domain. *Nat. Struct. Biol.*, 5(8):714–720, 1998.

[45] Harald Günther. *NMR spectroscopy: basic principles, concepts and applications in chemistry*. John Wiley & Sons, New York, 3rd edition, 2013.

[46] D. Hsu, T. Jiang, J.H. Reif, and Z. Sun. Bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4420–4426. IEEE, 2003.

[47] D. Hsu, J-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, pages 495–517, 1999.

[48] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3874–3880, Barcelona, Spain, 2005. IEEE.

[49] Sophie E. Jackson, Nadial elMasry, and Alan R. Fersht. Structure of the hydrophobic core in the transition state for folding of chymotrypsin inhibitor 2: A critical test of the protein engineering method of analysis. *Biochemistry*, 32:11270–11278, 1993.

[50] D. J. Jacobs. Generic rigidity in three-dimensional bond-bending networks. *J. Phys. A: Math. Gen.*, 31:6653–6668, 1998.

[51] Sam Ade Jacobs, Nicholas Stradford, Cesar Rodriguez, Shawna Thomas, and Nancy M. Amato. A scalable distributed RRT for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 5088–5095, Karlsruhe, Germany, May 2013.

[52] Leonard Jaillet, Judy Hoffman, Jur van den Berg, Pieter Abbeel, Josep M. Porta, and Ken Goldberg. EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2011.

[53] Taeho Jo and Jianlin Cheng. Improving protein fold recognition by random forest. *BMC Bioinformatics*, 15(11):1–7, 2014.

[54] Taeho Jo, Jie Hou, Jesse Eickholt, and Jianlin Cheng. Improving protein fold recognition by deep learning networks. *Scientific Reports*, 5:103–112, 2015.

[55] Marcelo Kallmann, Amaury Aubel, Tolga Abaci, and Daniel Thalmann. Planning collision-free reaching motion for interactive object manipulation and grasping. *Computer Graphics Forum*, 22(3):313–322, September 2003.

[56] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 30:846–894, 2011.

[57] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. ACM Symp. Theory of Computing (STOC)*, pages 353–362, May 1995.

[58] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[59] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.

[60] KUKA Robotics Corporation. KUKA youBot store. http://www.youbot-store.com. Accessed: June 1, 2013.

[61] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *CoRR*, abs/1402.6028, 2014.

[62] John Kuszewski, G. Marius Clore, and Angela M. Gronenborn. Fast folding of a prototypic polypeptide: The immunoglobulin binding domain of streptococcal protein G. *Protein Sci.*, 3:1945–1952, 1994.

[63] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4 – 22, 1985.

[64] S.M. Larson, C.D. Snow, M. Shirts, and V.S. Pande. Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology. *Computational Genomics*, 2004.

[65] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[66] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400, May 2001.

[67] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *New Directions in Algorithmic and Computational Robotics*, pages 293–308. A. K. Peters, 2001. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.

[68] M. Levitt. A simplified representation of protein conformations for rapid simulation of protein folding. *J. Mol. Biol.*, 104:59–107, 1976.

[69] M. Levitt. Protein folding by restrained energy minimization and molecular dynamics. *J. Mol. Biol.*, 170:723–764, 1983.

[70] R. Li and C. Woodward. The hydrogen exchange core and protein folding. *Protein Sci.*, 8(8):1571–1591, 1999.

[71] Jyh-Ming Lien, S.L. Thomas, and N.M. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4439–4444, sept. 2003.

[72] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 825–832, Cambridge, Massachusetts, 2005. MIT Press.

[73] Caroline Louis-Jeune, Miguel A. Andrade-Navarro, and Carol Perez-Iratxeta. Prediction of protein secondary structure from circular dichroism using theoretically derived spectra. *Proteins: Structure, Function, and Bioinformatics*, 80(2):374–381, 2012.

[74] Jose C. Martínez and Luis Serrano. The folding transition state between SH3 domains is conformationally restricted and evolutionarily conserved. *Nat. Struct. Biol.*, 6(11):1010–1016, 1999.

[75] Andreas Matouschek, Luis Serrano, Elizabeth M. Meiering, Mark Bycroft, and Alan R. Fersht. The folding of an enzyme v. H/^{2}H exchange–nuclear magnetic resonance studies on the folding pathway of barnase: Complementarity to and agreement with protein engineering studies. *J. Mol. Biol.*, 224:837–845, 1992.

[76] S. Matysiak and C. Clementi. Minimalist protein model as a diagnostic tool for misfolding and aggregation. *J. Mol. Biol.*, 363(1):297–308, 2006.

[77] Leland Mayne. Hydrogen exchange mass spectrometry. In Zvi Kelman, editor, *Isotope Labeling of Biomolecules - Applications*, volume 566 of *Methods in Enzymology*, pages 335–356. Academic Press, Salt Lake City, 2016.

[78] Troy McMahon, Sam Jacobs, Bryan Boyd, Lydia Tapia, and Nancy M. Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

[79] Troy McMahon, Sam Ade Jacobs, Bryan Boyd, Lydia Tapia, and Nancy M. Amato. Local randomization in neighbor selection improves PRM roadmap quality. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 4441–4448, Algarve, Portugal, October 2012.

[80] Troy McMahon, Shawna Thomas, and Nancy M. Amato. Reachable volume RRT. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2977–2984, Seattle, Wa., May 2015.

[81] Kevin Molloy and Amarda Shehu. Biased decoy sampling to aid the selection of near-native protein conformations. In *BCB*, pages 131–138. ACM, 2012.

[82] Marco Morales, Samuel Rodriguez, and Nancy M. Amato. Improving the connectivity of PRM roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 4427–4432, September 2003.

[83] Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato. A machine learning approach for feature-sensitive motion planning. In *Algorithmic Foundations of Robotics VI*, Springer Tracts in Advanced Robotics, pages 361–376. Springer, Berlin/Heidelberg, 2005. (WAFR '04).

[84] Marco A. Morales A., Roger Pearce, and Nancy M. Amato. Metrics for analyzing the evolution of C-Space models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1268–1273, May 2006.

[85] Marco A. Morales A., Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3114–3119, April 2005.

[86] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Knot planning from observation. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3887–3892, sept. 2003.

[87] Larissa A. Munishkina and Anthony L. Fink. Fluorescence as a method to reveal structures and membrane-interactions of amyloidogenic proteins. *Biochimica et Biophysica Acta (BBA) - Biomembranes*, 1768(8):1862–1885, 2007. Amyloidogenic ProteinMembrane Interaction.

[88] Victor Muñoz, Eric R. Henry, James Hoferichter, and William A. Eaton. A statistical mechanical model for β-hairpin kinetics. *Proc. Natl. Acad. Sci. USA*, 95(11):5872–5879, 1998.

[89] S. Nauli, B. Kuhlman, and D. Baker. Computer-based redesign of a protein folding pathway. *Nature Struct. Biol.*, 8(7):602–605, 2001.

[90] E. Plaku and L.E. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Algorithmic Foundations of Robotics VII*, Springer Tracts in Advanced Robotics, pages 3–18, Berlin/Heidelberg, 2006. Springer. (WAFR '06).

[91] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[92] David S. Riddle, Viara P. Grantcharova, Jed V. Santiago, Eric Alm, Ingo Ruczinski, and David Baker. Experiment and theory highlight role of native state topology in SH3 folding. *Nat. Struct. Biol.*, 6(11):1016–1024, 1999.

[93] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato. An obstacle-based rapidly-exploring random tree. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006.

[94] Samuel Rodriguez, Jyh-Ming Lien, and Nancy M. Amato. Planning motion in completely deformable environments. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2466–2471, May 2006.

[95] Samuel Rodriguez, Shawna Thomas, Roger Pearce, and Nancy M. Amato. (RESAMPL): A region-sensitive adaptive motion planner. In *Algorithmic Foundation of Robotics VII*, Springer Tracts in Advanced Robotics, pages 285–300. Springer, Berlin/Heidelberg, 2008. (WAFR '08).

[96] Yang Shen and Ad Bax. Protein backbone and sidechain torsion angles predicted from nmr chemical shifts using artificial neural networks. *Journal of Biomolecular NMR*, 56(3):227–241, 2013.

[97] M. S. Smyth and J. H. J. Martin. x ray crystallography. *Molecular Pathology*, 53(1):8–14, 2000.

[98] G. Song, S.L. Thomas, K.A. Dill, J.M. Scholtz, and N.M. Amato. A path planning-based study of protein folding with a case study of hairpin formation in protein G and L. In *Proc. Pacific Symposium of Biocomputing (PSB)*, pages 240–251, Lihue, HI, 2003. World Scientific.

[99] Sylvain Sorin. Exponential weight algorithm in continuous time. *Mathematical Programming*, 116(1-2):513–528, 2009.

[100] Lydia Tapia, Shawna L. Thomas, Bryan Boyd, and Nancy M. Amato. An unsupervised adaptive strategy for constructing probabilistic roadmaps. In

2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009, pages 4037–4044, 2009.

[101] Kaare Teilum, Birthe B. Kragelund, Jens Knudsen, and Flemming M. Poulsen. Formation of hydrogen bonds precedes the rate-limiting formation of persistent structure in the folding of ACBP. *J. Mol. Biol.*, 301:1307–1314, 2000.

[102] Shawna Thomas, Xinyu Tang, Lydia Tapia, and Nancy M. Amato. Simulating protein motions with rigidity analysis. In *Proceedings of the 10th annual international conference on Research in Computational Molecular Biology*, RECOMB'06, pages 394–409, Berlin, Heidelberg, 2006. Springer-Verlag.

[103] Shawna Thomas, Xinyu Tang, Lydia Tapia, and Nancy M. Amato. Simulating protein motions with rigidity analysis. *J. Comput. Biol.*, 14(6):839–855, 2007. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2006.

[104] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters,* 40(1991):175-179, 1991.

[105] Ana Rosa Viguera, Luis Serrano, and Matthias Wilmanns. Different folding transitions states may result in the same native structure. *Nat. Struct. Biol.*, 3(10):874–880, 1996.

[106] Virtudes Villegas, Jose C. Martínez, Francesc Z. Avilés, and Luis Serrano. Structure of the transition state in the folding process of human procarboxypeptidase A2 activation domain. *J. Mol. Biol.*, 283:1027–1036, 1998.

[107] Thomas E. Wales and John R. Engen. Hydrogen exchange mass spectrometry for the analysis of protein dynamics. *Mass Spec. Rev.*, 25(1):158–170, 2006.

[108] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc.*

IEEE Int. Conf. Robot. Autom. (ICRA), volume 2, pages 1024–1031, 1999.

[109] Hsin-Yi (Cindy) Yeh, Jory Denny, Aaron Lindsey, Shawna Thomas, and Nancy M. Amato. UMAPRM: Uniformly sampling the medial axis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 5798–5803, Hong Kong, P. R. China, June 2014.

[110] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *IEEE Trans. Robot. Automat.*, 23(1):151–157, 2007.

[111] Qian Yi and David Baker. Direct evidence for a two-state protein unfolding transition from hydorgen-deuterium exhchange, mass spectrometry, and NMR. *Protein Sci.*, 5:1060–1066, 1996.

[112] L. Zhang, Y.J Kim, and D. Manocha. A hybrid approach for complete motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7–14, 2007.

[113] Linxi Zhang, Jing Li, Zhouting Jiang, and Agen Xia. Folding rate prediction based on neural network model. *Polymer*, 44(5):1751–1756, 2003.